ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

İKİ ve ÜÇ BOYUTLU CİSİMLERİN VORONOİ DİYAGRAMLARININ ÇIKARILMASI ve DELAUNAY MOZAİKLEMESİNİN GERÇEKLEŞTİRİLMESİ

Bülent ÖZBEK

ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2010

TEZ ONAYI

Bülent ÖZBEK tarafından hazırlanan **"İki ve Üç Boyutlu Cisimlerin Voronoi Diyagramının Çıkarılması ve Delaunay Mozaiklemesinin Gerçekleştirilmesi"** adlı tez çalışması aşağıdaki jüri tarafından 26.02.2010 tarihinde oy birliği ile Ankara Üniversitesi Elektronik Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Yrd. Doç. Dr. Asım Egemen YILMAZ

Jüri Üyeleri :

Başkan: Yrd. Doç. Dr. Tolga CAN
Orta Doğu Teknik Üniversitesi,
Bilgisayar Mühendisliği Anabilim Dalı

Üye : Doç. Dr. Ziya TELATAR
Ankara Üniversitesi, Elektronik Mühendisliği Anabilim Dalı

Üye : Yrd. Doç. Dr. Asım Egemen YILMAZ
Ankara Üniversitesi, Elektronik Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. Orhan ATAKOL

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

İKİ VE ÜÇ BOYUTLU CİSİMLERİN VORONOİ DİYAGRAMLARININ ÇIKARILMASI VE DELAUNAY MOZAİKLEMESİNİN GERÇEKLEŞTİRİLMESİ

Bülent ÖZBEK

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Asım Egemen YILMAZ

Voronoi diyagramları ve Delaunay üçgenlemesi birçok bilim dalında kullanım alanı bulmuş önemli uygulamalı geometri araçlarıdır. Bu tez çalışmasında bu iki yapının arka planı incelenmekle beraber bu yapıların gerçek hayat problemlerinde kullanılabilmesi için geliştirilmiş algoritmalar ve bu yapıların çeşitli uygulamaları üzerinde durulmuş ve bu kapsamda bir yazılım geliştirilmiştir. Yazılımda 2 ve 3 boyutlu olmak üzere çeşitli Voronoi diyagramı ve Delaunay üçgenlemesi konularının görselleştirilmesi amaçlanmış ayrıca çeşitli algoritmaların (Fortune, Lawson ve Qhull algoritmaları) performans karşılaştırmaları yapılmıştır.

Şubat 2010, 91 sayfa

Anahtar Kelimeler: Voronoi Diyagramı, Delaunay Üçgenlemesi, Grafiksel Çifteşlik, İç Eksen, Alfa Şekilleri, Beta İskeletleri, Eğri ve Yüzey Modelleme

ABSTRACT

Master Thesis

VORONOI DIAGRAM EXTRACTION AND DELAUNAY TRIANGULATION FOR TWO AND THREE DMENSIONAL OBJECTS

Bülent ÖZBEK

Ankara University
Graduate School of Natural and Applied Sciences
Department of Electronics Engineering

Supervisor: Asst. Prof. Dr. Asım Egemen YILMAZ

Voronoi diagram and Delaunay triangulation are the important computational geometry tools that take place in a very large scale application fields in many sciences. In this thesis, the analysis of the background of these two structures as well as developed algorithms for use in real life problems, their various applications are taken into consideration, and a software has been developed within this concept. In the software, some practical applications of two and three dimensional Voronoi diagrams and Delaunay triangulations have been aimed to be visualized. Moreover, the runtime performance comparisons of various algorithms (Fortune, Lawson, and Qhull algorithms) have been performed.

February 2010, 91 pages

Key Words : Voronoi Diagram, Delaunay Triangulation, Graphical Duality, Medial Axis, Alpha Shapes, Beta Skeletons, Curve and Surface Reconstruction.

TEŞEKKÜR

Öncelikle tezimin yürütücülüğünü üstlenen, her türlü destek ve anlayışı gösteren, çalışmalarıma ışık tutan değerli hocam sayın Yrd. Doç. Dr. Asım Egemen YILMAZ'a (Ankara Üniversitesi Elektronik Mühendisliği Anabilim Dalı) en içten teşekkürlerimi sunarım.

Ayrıca, desteklerini esirgemeyen Elektronik Mühendisliği Bölümü'ndeki hocalarıma ve arkadaşlarıma, Kastamonu Üniversitesi Kastamonu M.Y.O.'daki çalışma arkadaşlarıma, her türlü sıkıntıma katlanan değerli aileme teşekkürü bir borç bilirim.

Bülent ÖZBEK
Ankara, Şubat 2010

İÇİNDEKİLER

SİMGELER DİZİNİ

$Vor(P)$	P noktalar kümesinin Voronoi Diyagramı
$Del(P)$	P noktalar kümesinin Delaunay Üçgenlemesi
$V(p)$	p noktasının Voronoi hücresi
$C_P(q)$	P noktalar kümesindeki q noktasının en büyük boş çemberi
$DG(P)$	P noktalar kümesinin Delaunay Grafiği
$DCEL$	Çift bağlantılı kenar listesi
$Vor(P,L)$	P noktalar kümesinin L doğru parçaları kümesi ile kısıtlanmış Voronoi diyagramı
$\alpha(P)$	P noktalar kümesinin α-şekli
$\beta(P)$	P noktalar kümesinin β-iskeleti

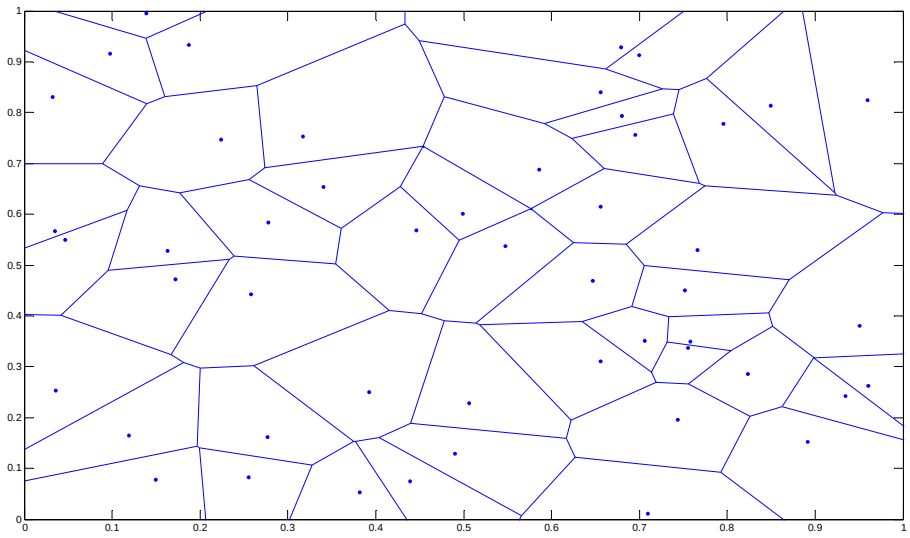
ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

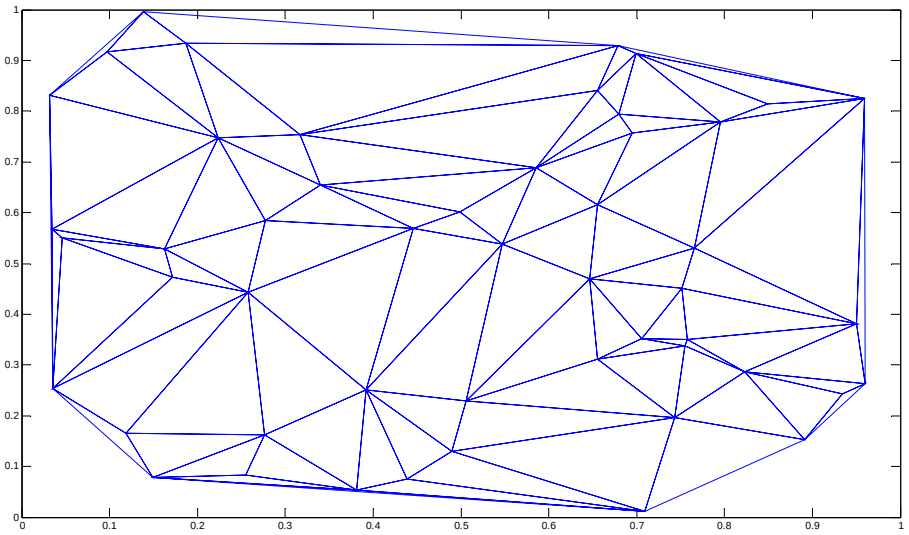
1. GİRİŞ

Voronoi diyagramı ve Delaunay üçgenlemesi hesaplamalı geometri alanının önemli hesaplama araçlarıdır. Kullanımları üç boyutlu modellemeden yüzey örgüsü üretimine, eşitlik gösteriminden alan hesaplamalarına kadar oldukça geniş alanlara yayılmıştır. Örneğin Voronoi diyagramları, çalışılan uzayı alt bölgelere ayırarak, uzamsal veri işleme ve yüzey modelleme gibi alanlarda çalışma imkanı sağlar (Key 2007). Delaunay üçgenlemesi ise yüksek kaliteli yüzey örgülerinin elde edilmesine olanak verir. Birçok çok boyutlu problem bu sayede grafikler ve yüzey örgüleri ile ifade edilerek nümerik bir çözüme daha kolay ulaşılmasını sağlar.

Şekil 1.1 ve 1.2'de iki boyutlu düzlemde alınmış 50 nokta için Voronoi diyagramı ve Delaunay üçgenlemesi gösterilmiştir.



Şekil 1.1 Rastgele alınmış 50 nokta için Voronoi diyagramı



Şekil 1.2 Rastgele alınmış 50 nokta için Delaunay üçgenlemesi

1.1 Uygulama Örnekleri

- Yangın gözlemleme kuleleri

Yangın gözlemleme kulelerinin olduğu sıkı bir orman olduğu düşünülsün. Her yangın gözlemleme kulesi kendine diğer kulelerden daha yakın ağaçlarda çıkabilecek bir yangını söndürmekten sorumludur. Herhangi bir kulenin sorumlu olduğu ağaçların kümesi o küme için Voronoi hücresi ya da Voronoi poligonu ile bulunabilir. Voronoi diyagramı sorumluluk bölgeleri arasındaki sınırları oluşturur.

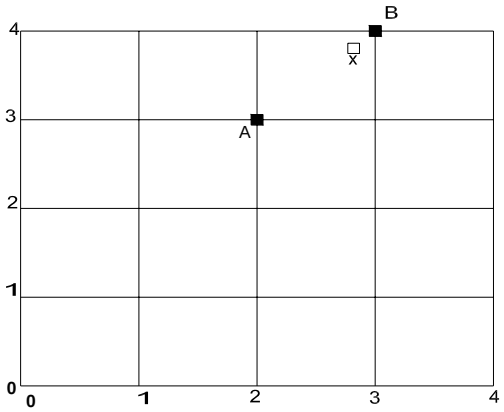
- Yanan kuleler

Şimdi de tersi bir durum oluşsun. Bütün kuleler aynı anda tutuşturulsun ve orman eşit oranda yanmaya başlasın. Yangın, merkezi kuleler olan daireler şeklinde yayılacaktır ve yangının söneceği yer iki ya da daha fazla kuleden eşit uzaklıkta olan Voronoi köşeleri veya kenarları olacaktır.

- En yakın komşu kümelemesi

Örüntü tanıma alanında yaygın kullanım alanı bulan bir teknik hedef nesneleri, koordinatları özellik ölçümleri olacak şekilde noktalara indirgenerek bir özellik uzayına haritalanmasıdır. Böylelikle bilinmeyen bir nesne özellik uzayında kendine en yakın hedef nesne ile tanımlanabilir (O'Rourke 1997).

Örneğin içinde A ve B adında iki tür vida somunu bulunan bir parça kutusu düşünülsün. A'nın iç ve dış çapları sırasıyla 2cm ve 3cm ve B'nin iç ve dış çapları sırasıyla 3cm ve 4cm'dir. Özellik uzayı Öklit düzleminin pozitif dördünü olacaktır çünkü çaplar negatif olamaz. A somunu (2,3), B somunu da (3,4) noktalarına düşürülür.



Şekil 1.3 Özellik uzayı örneği

Bir görüntüleme sistemi bu kutudaki *x* parçasından görüntü almış ve iç ve dış çapları 2.8cm ve 3.7cm olarak ölçmüştür. Ölçme işleminde kesinlik olmadığı bilinmektedir ve kutuda sadece A ve B tipinde somunlar vardır. Öyleyse bu *x* somununun A veya B tipinde olması gereklidir. Bu parça B tipinde olmalıdır çünkü özellik uzayında bu parçanın A'ya uzaklığı 1.06 iken B'ye uzaklığı 0.36'dır. Diğer bir söylemle *x*'in en yakın komşusu B'dir. Çünkü *x*, B'nin Voronoi hücresi içinde yer almaktadır. (Şekil 1.3)

- Tesis Konumu

Rakip birçok marketin olduğu bir bölgede yeni bir market açılmak istensin. Nüfus yoğunluğunun düzgün dağılımlı olduğu kabul edilerek yeni marketin satışlarının optimum olması için yerinin doğru belirlenmesi gereklidir. Bu durumu çözmenin en akla yatkın yolu eski marketlerden mümkün olduğunca uzak bir noktaya konumlanmak olacaktır yani yeni market kendine en yakın olacak markete mümkün olduğunca uzakta olmalıdır. Bu da marketi, en büyük boş çemberin merkezine yerleştirmekle mümkündür. En büyük boş çemberin içinde başka market olmayacaktır. Böylelikle en yakın markete olan uzaklık çemberin yarıçapı kadardır.

- Güzergâh planlama

Bir robotun geçmesi gereken engelli bir yol olsun. Robotun çarpma riskini en aza indirmek için robot yoldaki bütün nesnelerden olabildiğince uzak noktalardan geçmelidir. Eğer problem iki boyutla sınırlandırılırsa ve robotun dairesel yapıda olduğu düşünülürse robotun, nesnelerin Voronoi diyagramının üzerinde dolaşması yeterli olacaktır. Eğer nesneler nokta gibi düşünülürse bu klasik Voronoi diyagramı olacaktır. Aksi takdirde yani nesneler poligon ya da daha karmaşık yapıda şekillere sahipse Voronoi diyagramının genelleştirilmiş bir versiyonu doğru yolun çıkarılmasını sağlayacaktır.

- Kristal bilim

Belli sayıda kristal çekirdeğinin sabit hızda düzenli olarak büyüdüğü düşünülsün. Artık büyümenin mümkün olmadığı zaman kristalin görünümü nasıl olacaktır? Bunu bulmak için Voronoi hücrelerinin bulunması yeterlidir. Çünkü her çekirdek ancak kendi Voronoi hücresinin sınırına kadar büyüyebilecek sonra komşu çekirdeğin bölgesine dayanacaktır.

1.2 Teze Yaklaşım

Bu tez çalışmasında hesaplamalı geometri alanının iki önemli aracı olan Voronoi diyagramı ve Delaunay üçgenlemesi ile bunların çeşitli genelleştirilmeleri detaylı bir şekilde incelenmeye çalışılmış, bu yapıların hesaplanmasında kullanılan çeşitli algoritmalarda araştırılmıştır. Bu algoritmaları kullanarak Voronoi diyagramını ve Delaunay üçgenlemesini hesaplayan ve bu yapıları görselleştirme imkanı sunan bir yazılımın geliştirilmesi amaçlanmıştır. Bu yazılım ile tezde incelenen bazı genelleştirilmiş Voronoi diyagramı ve Delaunay üçgenlemesi konularının görselleştirilmesi de hedeflenmiştir.

Bu kapsamda tezin 2. bölümünde Voronoi Diyagramı ve Delaunay üçgenlemesi tanıtılarak matematiksel alt yapısı ve temel bazı özellikleri incelenmiştir. Yine bu bölümde birbirlerine grafiksel anlamda sıkı bir bağlantısı olan bu iki yapının çifteşliği ortaya konulmuştur. Tezin 3. bölümünde bu yapıların hesaplanması için geliştirilmiş çeşitli algoritmalar sunulmuş ve karşılaştırmaları yapılmıştır. Ayrıca gerçek hayatta uygulama alanı bulan bazı genelleştirilmiş Voronoi diyagramı ve Delaunay üçgenlemesi konuları da incelenmeye çalışılmıştır.

4. ve 5. bölümlerde tez kapsamında geliştirilen VVD3 yazılımı hakkında genel bilgi verilmiş ve bu yazılımla elde edilen çeşitli sonuçlar ortaya konulmuştur. Tezin son bölümünde ise bu çalışmanın genel bir değerlendirilmesi ve gelecek çalışmaların ne olabileceği sunulmuştur.

2. KURAMSAL TEMELLER

Voronoi diyagramı, Delaunay üçgenlemesi ve bu iki yapı arasındaki ilişkinin incelendiği bu bölümde konu bütünlüğü ve notasyon uyumluluğu sağlanması amacıyla de Berg, Cheong, van Kreveld ve Overmars tarafından kaleme alınmış olan "Computational Geometry – Algorithms and Applications" adlı kitabın yedinci, sekizinci ve dokuzuncu bölümleri takip edilmiştir.

2.1 Voronoi Diyagramları

2.1.1 Giriş

Voronoi mozaiklemesi veya Dirichlet mozaiklemesi olarak da bilinen Voronoi diyagramı ayrık noktalar kümeleri arasındaki uzaklıklarla tanımlanan, metrik uzayın özel bir parçalanması yöntemidir. Bu yöntemin kökeni 17. yüzyıla kadar uzanmaktadır. R. Descartes çizimlerinde uzayı, merkezinde yıldızların olduğu konveks bölgelere parçalayarak göstermiştir. Dirichlet (1850) ve Voronoi (1908, 1909) adındaki matematikçiler bu yapıyı formal olarak tanımlayan ilk bilim adamlarıdır.

Bir noktalar kümesinin Voronoi diyagramı düzlem veya uzayın her nokta için parçalara ayrılmasıdır. Bu parçalar düzlemin o noktaya, diğer bütün noktalardan daha yakın kısmını içerir.

2.1.2 Tanım ve temel özellikler

"p" ve "q" olarak isimlendirilen iki nokta arasındaki Öklit uzaklığı $dist(p,q)$ ile gösterilsin. Düzlemde bu uzaklık;

$$dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{2.1}$$

ile tanımlanır.

P'nin düzlemde n adet ayrık noktadan oluşan bir küme olduğu düşünülsün. Bu noktalar düzlemdeki mevzilerdir.

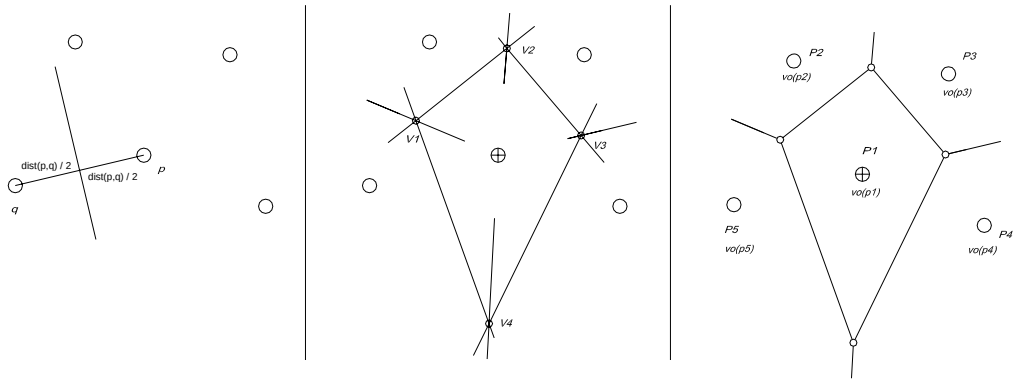
$$P = \{p_1, p_2, \dots\dots\dots, p_n\}$$

P'nin Voronoi diyagramı düzlemin, her mevzi için bir tane olmak üzere n adet alt bölgeye ayrılması ile elde edilir. Her alt bölge için, herhangi bir q noktası sadece ve sadece

$$dist(q, p_i) < dist(q, p_j); \qquad p_j \in P, \ j \neq i \tag{2.2}$$

kuralını sağlıyorsa p_i hücresi içindedir.

P'nin Voronoi Diyagramı $Vor(P)$ ile gösterilir. Terminolojinin biraz daraltılması ile çoğu zaman $Vor(P)$ veya Voronoi Diyagramı sadece alt bölgelerin kenar ve köşelerini ifade eder. Örneğin, bağlı Voronoi Diyagramı'ndan kastedilen, diyagramın kenar ve köşelerinin birleşiminin bağlı bir küme oluşturmasıdır. $Vor(P)$'nin p_i'ye karşılık gelen hücresi $V(p_i)$ ile gösterilir ve p_i'nin Voronoi hücresi olarak ifade edilir. (Şekil 2.1)



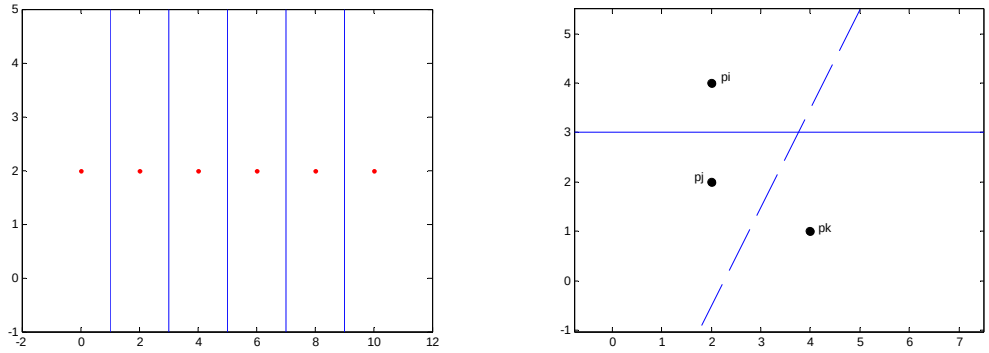
Şekil 2.1 $\boldsymbol{p}$ ve $\boldsymbol{q}$ noktalarının: a. dik açıortayı, b. Voronoi köşeleri c. Voronoi hücreleri

Öncelikle tek bir Voronoi hücresinin yapısının incelenmesi doğru olacaktır. Düzlemdeki p ve q noktalarının açıortayı, $\overline{pq}$ doğru parçasını iki eşit parçaya bölen doğru olarak tanımlanabilir. Bu açıortay, düzlemi iki yarı-düzleme böler. Burada p'yi içeren açık yarı-düzlem $h(p,q)$ ve q'yu içeren açık yarı-düzlem $h(q,p)$ ile ifade edilsin. Buradan,

$$V(p_i) = \cap_{1 \le j \le n, j \ne i}\, h(p_i, q_i) \qquad (2.3)$$

Burada da görüldüğü gibi $V(p_i)$, $n-1$ yarı-düzlemin kesişimidir ve en fazla $n-1$ köşe ve $n-1$ kenarla sınırlandırılmış (sınırlandırılmamış da olabilir) açık konveks poligon bölgesidir.

Bu Voronoi hücrelerinin birleşmesi ile Voronoi diyagramının tamamı oluşur. Diyagramın her Voronoi hücresinin çeşitli sayıda yarı-düzlemin kesişmesi sonucu oluştuğu göz önüne alınırsa, Voronoi diyagramı düzlemin kenarları düz olan parçalara ayrılmasıdır. Eğer bütün noktalar doğrusal değilse kenarlar yarı-doğru ya da doğru parçasıdır (Şekil 2.2).



Şekil 2.2.a Paralel doğruların Voronoi diyagramı, b. Kesişen doğruların Voronoi diyagramı

Teorem 2.1: P'nin düzlemdeki noktaların kümesi olduğu düşünülsün. Eğer bütün noktalar doğrusal ise $Vor(P)$ $n-1$ adet paralel doğrudan oluşur (Şekil 2.2.a). Aksi takdirde $Vor(P)$ bağlantılıdır ve kenarları doğru parçalarından veya yarı-doğrulardan oluşur (Şekil 2.2.b).

İspat: Öncelikle P kümesindeki noktaların doğrusal olmaması durumunda $Vor(P)$'nin kenarlarının doğru parçası ya da yarı-doğru olduğu gösterilecektir. $Vor(P)$'nin kenarlarının tam doğru olduğu bilinmiyor. Şimdi, $Vor(P)$'nin "e" olarak adlandırılan bir kenarının tam doğru olduğu şeklinde bir karşıtlık ortaya atılsın. e doğrusu $V(p_i)$ ve $V(p_j)$ hücrelerinin kenarını oluştursun. $p_k \in P$ noktası da p_i ve p_j noktaları ile doğrusal olmayan bir nokta olsun. p_j ve p_k noktalarının açıortayı e doğrusuna paralel değildir ve bu yüzden e doğrusunu keser. Bu durumda e'nin $h(p_k, p_j)$ hücresi içinde kalan kısmı $V(p_i)$'nin sınırı olamaz, çünkü bu parça p_k' ya p_j'den daha yakındır. Dolayısıyla ortaya atılan karşıtlık çürümüş olur. Buna bağlı olarak yukarıda söylenilen gibi noktalar aynı doğrultuda olmadığı sürece Voronoi diyagramının kenarları tam doğru olamaz; doğru parçası ya da yarı-doğrudur.

Şimdi de $Vor(P)$'nin bağlantılı olduğunu gösterilmeye çalışılsın. Eğer $Vor(P)$ bağlantılı olmasaydı, o zaman düzlemi ikiye bölen bir $V(p_i)$ hücresinin olması gerekirdi. Voronoi hücreleri konveks olduklarından $V(p_i)$ iki paralel tam doğru tarafından sınırlandırılmış bir bölge olmalıydı. Ancak az önce gösterildiği gibi Voronoi

diyagramının kenarları tam doğru olamayacakları için Voronoi diyagramı bağlantılı olmalıdır.

Buraya kadar Voronoi diyagramının yapısı incelenmiştir. Şimdi de diyagramın karmaşıklığı araştırılacaktır. Buradaki amaç diyagramın toplam kenar ve köşe sayısını belirlemek olacaktır.

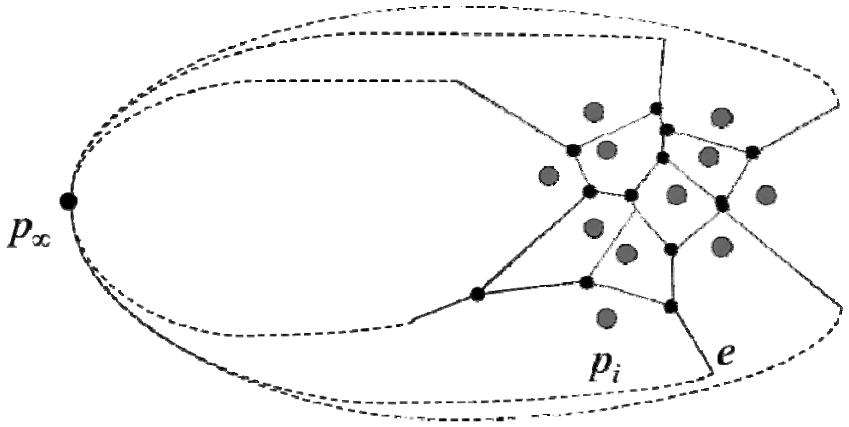
n adet Voronoi mevzisi olduğundan ve her Voronoi hücresi en fazla $n-1$ kenar ve köşeye sahip olduğundan $Vor(P)$'nin karmaşıklığı en fazla ikinci dereceden olabilir. $Vor(P)$'nin gerçekten ikinci dereceden karmaşıklığa sahip olduğu kesin değildir. Bir Voronoi hücresinin birinci dereceden karmaşıklığa sahip olduğu bir örnek oluşturmak son derece kolaydır. Fakat birçok Voronoi hücresi birinci dereceden karmaşıklığa sahip olabilir mi? Aşağıdaki teorem bunun mümkün olmadığını ve Voronoi hücrelerinin ortalama köşe sayısının 6 olduğunu göstermektedir.

Teorem 2.2: $n \geq 3$ için düzlemdeki n adet mevzinin Voronoi diyagramındaki köşe sayısı en fazla $(2n-5)$ ve kenar sayısı en fazla $(3n-6)$'dır.

Bu teoremin ispatında Euler'in formülü kullanılır. Bu formülde herhangi bir bağlantılı düzlemsel grafik için m_v köşeleri, m_e ayrıtları(kenarları) ve m_f yüzleri ifade etmek üzere

$$m_v - m_e + m_f = 2 \qquad (2.4)$$

olarak tanımlanmıştır. Euler'in formülü doğrudan Voronoi diyagramına uygulanamaz çünkü $Vor(P)$ yarı doğrular da içermektedir ve bu yüzden düzgün bir grafik değildir. Bu durumu aşmak için Voronoi diyagramına p_∞ adı verilen sonsuzda bir nokta ilave edilir ve bütün yarı doğrular bu köşe noktasında birleştirilir. (Şekil 2.3) Böylelikle düzgün bir grafik elde edilmiş olur ve Euler'in formülü Voronoi diyagramına uygulanabilir.



Şekil 2.3 Voronoi diyagramına noktasının eklenmesi

'nin köşelerini, 'nin kenarlarını ve de mevzi sayısını ifade etme üzere

– (2.5)

bağıntısı ele edilir.

Genişletilmiş bir grafikte her kenarın tam olarak iki köşe noktasına sahip olması gerekir. Dolayısıyla eğer köşe noktalarının dereceleri toplanırsa kenar sayısının iki katı elde edilir. köşe noktası da dahil olmak üzere her köşenin de derecesinin en az 3 olması gerektiği düşünülürse aşağıdaki eşitsizlik elde edilir.

(2.6)

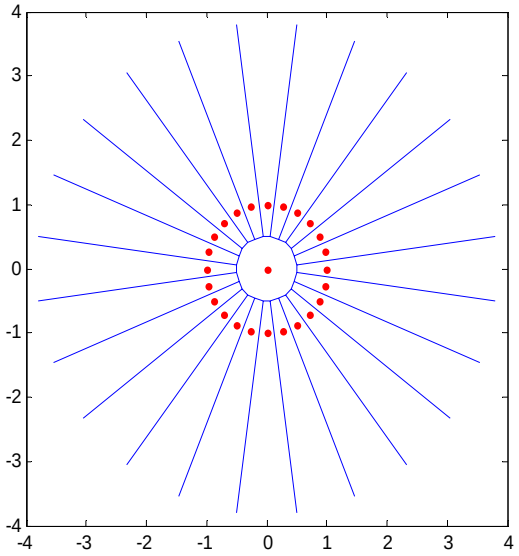
2.5 ve 2.6 ifadeleri kullanılarak

– (2.7a)

ve

$$n_v \leq 2n - 5 \tag{2.7b}$$

sonuçları elde edilebilir (Şekil 2.4).



Şekil 2.4 Voronoi diyagramının kenar ve köşeleri

Son olarak Voronoi diyagramının kenar ve köşelerinin özellikleri ortaya konulacaktır. Kenarların iki nokta arasındaki çizginin dik açıortayının belirli bir bölümü ve köşelerin de bu açıortayların kesişim noktaları oldukları biliniyor. $Vor(P)$ sadece lineer iken bu açıortayların sayısı ikinci derecedir. Bu yüzden her açıortay Voronoi diyagramının kenarlarını tanımlamaz ve her kesişim noktası da Voronoi diyagramının köşesi değildir. Hangi kesişim noktalarının ve hangi açıortayların Voronoi diyagramını oluşturduğunu ortaya koymak için şu tanımlama yapılabilir.

Bir q noktasının P için "en büyük boş çemberi" $C_P(q)$ ile gösterilir ve merkezinde q'nun olduğu ve P'nin başka hiçbir mevzisini içermeyen en büyük çember olarak tanımlanır. Buna bağlı olarak aşağıdaki teorem Voronoi diyagramının kenar ve köşelerini karakterize eder.

Teorem 2.3: Bir P noktalar kümesinin Voronoi diyagramı $Vor(P)$ için aşağıdaki iki ifade geçerli olmalıdır.

i) Bir q noktası sadece ve sadece o noktanın en büyük boş çemberi $C_P(q)$ sınırlarında 3 veya daha fazla mevzi barındırıyorsa $Vor(P)$'nin köşe noktasıdır.

ii) p_i ve p_j noktaları arasındaki açıortay, sadece ve sadece eğer en büyük çemberi $C_P(q)$ üzerinde p_i ve p_j bulundurup başka hiçbir mevzi içermeyen bir q noktası varsa Voronoi diyagramının kenarıdır.

$C_P(q)$ en büyük boş çemberi üzerinde 3 veya daha fazla mevzinin bulunduğu bir q noktası olduğu düşünülsün. p_i, p_j ve p_k bu mevzilerden üçü olsun. $C_P(q)$ çemberinin içi boş olduğundan q noktası $V(p_i)$, $V(p_j)$ ve $V(p_k)$'nın sınırında olmalıdır ve bu yüzden q noktası $Vor(P)$'nin bir köşe noktasıdır.

Diğer yandan $Vor(P)$'nin her q köşe noktası en az 3 kenar ile, dolayısıyla da $V(p_i)$, $V(p_j)$ ve $V(p_k)$ ile temas etmelidir. q noktasının p_i, p_j ve p_k mevzilerine eşit uzaklıkta olması gerektiğinden başka hiçbir mevzi q noktasına bu mevzilerden daha yakın olamaz. Aksi takdirde $V(p_i)$, $V(p_j)$ ve $V(p_k)$ q noktasında buluşamaz. Buradan da p_i, p_j ve p_k mevzilerinin bulunduğu çemberin içinde başka hiçbir mevzi olamaz.

Teoremin ikinci kısmında anlatılan özelliğe sahip bir q noktası olduğu düşünülsün. $C_P(q)$ çemberi içinde mevzi olmadığından ve çember üzerinde p_i ve p_j noktaları olduğundan

$$dist(q, p_i) = dist(q, p_j) \leq dist(q, p_k)\ 1 \leq k \leq n \qquad (2.8)$$

olmalıdır. Bu, q noktasının $Vor(P)$'nin bir köşe noktası olduğunu ya da bir kenarı üzerinde bulunduğunu gösterir. Teoremin birinci kısmı q'nun $Vor(P)$'nin bir köşe

noktası olamayacağını gösterir. Buradan da q'nun $Vor(P)$'nin p_i ve p_j mevzilerinin dik açıortayı ile tanımlanan kenarı üzerinde olduğu sonucuna varılır.

Tersine düşünülürse, p_i ve p_j'nin dik açıortayının bir Voronoi kenarı tanımladığı kabul edilsin. Bu kenar üzerindeki herhangi bir q noktasının en büyük boş çemberi üzerinde sadece ve sadece p_i ve p_j mevzileri olmalı ve başka hiçbir mevzi bulunmamalıdır.

2.2 Delaunay Üçgenlemesi

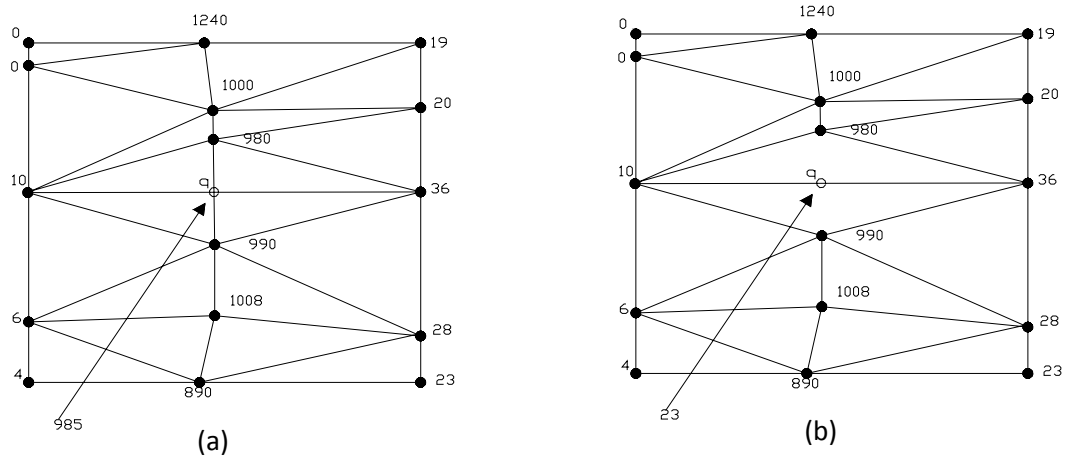
2.2.1 Giriş

1934 yılında Delaunay tarafından bulunan Delaunay Üçgenlemesi yüzey örgüleri ile çalışırken çok yararlı bir araçtır (Delaunay 1934). Bir $P = \{p_1, p_2, \dots\dots\dots, p_n\}$ noktalar kümesi için Delaunay üçgenlemesi $Del(P)$ ile gösterilir. En önemli avantajlarından biri verilen bir noktalar kümesinin bütün üçgenlemeleri boyunca minimum açıyı maksimize etmesidir. Bir üçgenleme, üçgenlemenin tüm kenar ve üçgenleri için Delaunay kriterini sağlıyorsa Delaunay üçgenlemesidir. Bu kriter sağlandığında üçgenleme en büyük minimum açıya sahip olur. Bu özellik yüksek kaliteli yüzey örgüleri sağlar.

2.2.2 Üçgenleme

Üçgenleme işlemi çok değişik yöntemlerle yapılabilir. Fakat hangi üçgenlemenin amaca en uygun olduğuna karar verilmelidir. Üçgenleme yapılacak yüzeyin orijinal hali bilinmemektedir ve sadece örneklenen noktaların yükseklik bilgisi vardır. Başka hiçbir bilgi olmadığına göre ve bütün üçgenlemelerde aynı bilgi kullanılacağına göre bütün üçgenlemelerin eşit derecede yüzeyi ifade etmesi gerekir. Oysa bazı üçgenlemeler daha doğal görünür. Örneğin Şekil 2.5'te aynı noktalar kümesinin iki farklı üçgenlemesi gösterilmiştir. Yükseklik değerlerine bakıldığında bu üçgenlemenin bir dağ sırasına ait olduğu düşünülebilir. Şekil 2.5.a'da verilen üçgenleme bunu hissettirmektedir ancak Şekil 2.5.b'deki üçgenlemede sadece tek bir kenarın çevrilmesiyle, dağ sırasını kesen

bir vadinin olduğu hissini vermektedir. Oysa bu doğal görünmemektedir. Bu duruma bakılarak Şekil 2.5.a üçgenlemesinin Şekil 2.5.b'den daha iyi bir üçgenleme olduğunu belirten bir kriter geliştirilebilir.



Şekil 2.5 Verilen noktalar kümesinin iki farklı üçgenlemesi

Şekil 2.5.b'deki problem q noktasındaki yüksekliğin birbirinden göreceli olarak uzakta iki nokta ile belirlenmesidir. Bu olay, q noktasının iki uzun ve keskin üçgenin ortasında yer almasından kaynaklanmaktadır. Bu üçgenlerin inceliği soruna neden olmaktadır. Dolayısıyla küçük açılara sahip üçgenlemeler kötü sonuç vermektedir. Bu yüzden üçgenlemeler sahip oldukları en küçük açıya göre sınıflandırılabilir. Eğer iki üçgenlemenin en küçük açıları eşitse, ikinci en küçük açıya bakılır ve bu şekilde devam edilir. Verilen bir P noktalar kümesinin sadece sınırlı sayıda üçgenlemesi olduğundan minimum açıyı maksimize eden bir üçgenleme olmalıdır.

$P = \{p_1, p_2, \dots\dots\dots, p_n\}$ düzlemde bir noktalar kümesi olsun. P'nin bir üçgenlemesinin tanımlanabilmesi için önce azami düzlemsel bölümleme tanımı yapılmalıdır. Azami düzlemsel bölümleme, düzlemselliği bozulmadan iki noktayı birleştiren bir nokta eklenemeyecek bölümleme olarak tanımlanabilir. Diğer bir deyişle, S bölümlemesi içinde olmayan herhangi bir kenar daha önceki kenarlardan birini keser. Buna bağlı olarak şimdi P'nin üçgenlemesi, köşe kümesi P olan azami düzlemsel bölümleme olarak tanımlanabilir.

Bu tanıma göre bir noktalar kümesi için kesinlikle bir üçgenleme vardır. Ancak bu üçgenleme sadece üçgenlerden mi oluşur? Evet, sınırlandırılmamış yüzler hariç bütün yüzler üçgen olmalıdır; sınırlandırılmış bir yüzey poligon olduğuna göre herhangi bir poligon üçgenlenebilir. P'nin konveks zarf sınırındaki iki noktayı birbirine bağlayan doğru parçası da P'nin herhangi bir üçgenlemesinin kenarıdır. Buna bağlı olarak T üçgenlemesinin sınırlandırılmış yüzlerinin birleşimi her zaman P'nin konveks zarfıdır ve sınırlandırılmamış yüz de konveks zarfın tümleyenidir.

P'nin bütün üçgenlemelerindeki üçgen sayısı aynıdır. Bu kural üçgenlemelerin kenar sayısı için de geçerlidir. Üçgen ve kenarların tam sayısı P'nin konveks zarf sınırındaki noktaların sayısına bağlıdır.

Teorem 2.4: P düzlemde aynı doğru üzerinde olmayan noktaların kümesi olsun. P'nin konveks zarf sınırındaki noktalarının sayısı da k ile gösterilsin. Bu durumda P'nin herhangi bir üçgenlemesi $(2n - 2 - k)$ üçgenden ve $(3n - 3 - k)$ kenardan oluşur.

İspat : T P'nin bir üçgenlemesi olmak üzere m T'nin üçgen sayısı olsun. T üçgenlemesinin yüz sayısı $n_f = m + 1$ olur. Her üçgen üç kenara ve sınırlandırılmamış yüz ise k kenara sahiptir. Ayrıca her kenar iki yüz tarafından paylaşılmaktadır. Bu yüzden T'nin toplam kenar sayısı

$$n_e = (3m + k) / 2 \tag{2.9}$$

olacaktır.

Euler'in formülüne göre

$$n - n_e + n_f = 2$$

olduğuna göre, n_e ve n_f değerleri formülde yerine yazılırsa;

$$m = 2n - 2 - k \tag{2.10}$$

ve

$$n_e = 3n - 3 - k \tag{2.11}$$

elde edilir.

P'nin bir üçgenlemesi olan T'nin m adet üçgeni olduğu düşünülsün. T'nin üçgenlerinin $3m$ adet açısının artan sırada dizilsin. $\alpha_1, \alpha_2, \ldots\ldots, \alpha_{3m}$ bu şekilde düzenlenmiş sıralama olsun. Burada her $i < j$ için $\alpha_i \leq \alpha_j$ 'dir. $A(T) = (\alpha_1, \alpha_2, \ldots\ldots, \alpha_{3m})$ T'nin açı vektörü olarak adlandırılır. (T')'de P'nin diğer bir üçgenlemesi olsun.

$A(T') = (\alpha_1', \alpha_2', \ldots\ldots, \alpha_{3m}')$ de (T') üçgenlemesinin açı vektörü olur. Eğer $1 \leq i \leq m$ aralığında $j < i$ için,

$$\alpha_j = \alpha_j' \tag{2.12}$$

ve

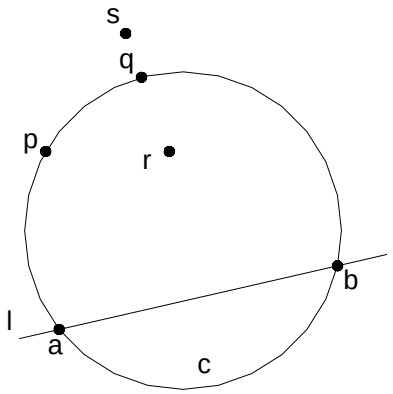
$$\alpha_i > \alpha_i' \tag{2.13}$$

kurallarını sağlayan bir i indeksi varsa T'nin açı vektörü (T') nün açı vektöründen büyüktür denir ve $A(T) > A(T')$ ile gösterilir. Eğer P'nin diğer bütün (T') üçgenlemeleri için $A(T) \geq A(T')$ ise T üçgenlemesine optimal açılı üçgenleme denir. Optimal açılı üçgenlemeler örnek noktalar kümesinden çok yüzlü bir yüzey modeli elde edilmek istendiğinde en iyi sonucu verir. Aşağıdaki teoremde bir üçgenlemenin ne zaman optimal açılı olduğu gösterilmiştir. Bu teorem Thales teoremi olarak da bilinmektedir. Burada p, q, r ile gösterilen üç noktayla tanımlanan en küçük açı $\forall pqr$ ile gösterilmiştir.

Teorem 2.5: C bir çember olsun. l doğrusu C çemberini a ve b noktalarında kesen bir doğru olsun. p, q, r ve s de l'nin aynı tarafında bulunan noktalar olsun. p ve q C üzerinde, r çemberin içinde ve s çemberin dışında yer alsın (Şekil 2.6). Bu durumda

$$\measuredangle arb > \measuredangle apb = \measuredangle aqb > \measuredangle asb \tag{2.14}$$

olacaktır.

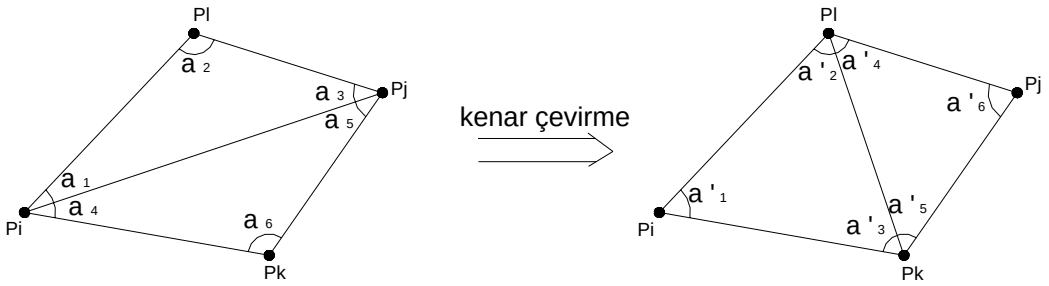


Şekil 2.6 Bir çember üzerinde, içinde ve dışında noktalar

$e = \overline{p_i p_j}$ P'nin T üçgenlemesinin bir kenarı olsun. Eğer e sınırlandırılmamış yüzün bir kenarı değilse $p_i p_j p_k$ ve $p_i p_j p_l$ üçgenlerinin ortak kenarıdır. Bu iki üçgen konveks bir dörtgen oluşturduğundan $\overline{p_i p_j}$ kenarını çıkarıp yerine $\overline{p_k p_l}$ kenarını koyarak (T') adında yeni bir üçgenleme elde edilebilir. Bu işleme kenar çevirme denir. Bu durumda T ve T' üçgenlemelerinin açı vektörleri arasındaki fark $A(T)$'deki $\alpha_1, \alpha_2, \dots, \alpha_6$ $A(T')$'nde $\alpha_1', \alpha_2', \dots, \alpha_6'$ olarak değiştirildiği için 6 açıdır. Eğer

$$\min_{1 \le i \le 6} \alpha_i < \min_{1 \le i \le 6} \alpha_i' \tag{2.15}$$

ise $\overline{p_i p_j}$ kenarı geçersiz bir kenar olarak adlandırılır. Diğer bir deyişle eğer bir kenar çevrilerek minimum açı artırılabiliyorsa o kenar geçersiz bir kenardır (Şekil 2.7).



Şekil 2.7 Kenar çevirme işlemi

T geçersiz bir e kenarı olan bir üçgenleme ve T' de T'den e'nin çevrilmesiyle elde edilen bir üçgenleme olsun. Bu durumda $A(T') > A(T)$ olacaktır. e kenarının geçerli olup olmadığının belirlenebilmesi için $\alpha_1, \alpha_2, \dots, \alpha_6$, ve $\alpha_1', \alpha_2', \dots, \alpha_6'$ açılarının hesaplanmasına gerek yoktur. Bunun yerine aşağıda belirtilen basit kriter geliştirilebilir.

$\overline{p_i p_j}$ kenarı $p_i p_j p_k$ ve $p_i p_j p_l$ üçgenlerinin ortak kenarı ve C'de $p_i,\ p_j,\ p_k$noktalarından geçen bir çember olsun. $\overline{p_i p_j}$ kenarı sadece ve sadece $p_l$noktası C çemberi içinde yer alıyorsa geçersiz bir kenardır. Ayrıca $p_i,\ p_j,\ p_k,\ p_l$ noktaları bir dörtgen oluşturuyorsa ve hepsi bir çemberin üzerinde yer almıyorsa $\overline{p_i p_j}$ ve $\overline{p_k p_l}$ kenarlarından biri kesinlikle geçersiz bir kenardır.

Bu kriter p_k ve p_l noktaları için simetriktir. Eğer p_l $(p_i,\ p_j,\ p_k)$ noktalarından geçen çemberin içindeyse p_k'da $(p_i,\ p_j,\ p_l)$ noktalarından geçen çemberin içindedir. Eğer dört noktanın hepsi aynı çemberin üzerinde ise $\overline{p_i p_j}$ ve $\overline{p_k p_l}$ kenarlarının her ikisi de geçerli kenarlardır. Ortak bir geçersiz kenarı olan iki üçgen konveks bir dörtgen oluşturduğundan her zaman kenar çevirme işlemi yapılabilir.

Geçerli bir üçgenleme geçersiz bir kenarı olmayan üçgenleme olarak tanımlanabilir. Herhangi bir optimal açılı üçgenleme geçerli bir üçgenlemedir. Bir üçgenleme verildiğinde geçerli bir üçgenlemenin hesaplanması oldukça basittir. Bütün kenarlar geçerli kenarlar olana kadar geçersiz kenarlara kenar çevirme işlemi uygulanır.

```
Algoritma GEÇERLİ_ÜÇGENLEME(T)

Girdi : P noktalar kümesinin herhangi bir üçgenlemesi

Çıktı : P'nin geçerli bir üçgenlemesi

   1. while (T geçersiz bir p_ip_j kenarı içerdiği sürece)

      do ( p_ip_j kenarını çevir)

      p_ip_jp_k ve p_ip_jp_l p_ip_j kenarını paylaşan iki üçgen iken

      p_ip_j kenarını çıkar ve  p_kp_l kenarını ekle

      return (T)
```

Algoritmanın her döngüsünde T'nin açı vektörünün değeri büyüyecektir. P'nin sonlu sayıda üçgenlemesi olduğundan algoritma sonlanacaktır. Algoritma sonlandığında P'nin geçerli bir T üçgenlemesi elde edilir.

2.2.3 Delaunay üçgenlemesi

P düzlemde bir noktalar veya mevziler kümesi olsun. P'nin Voronoi diyagramı P'nin her noktası veya mevzisi için, düzlemin, o mevziye diğer bütün mevzilerden daha yakın noktalardan oluşan bölgelere ayrılmasıdır. p mevzisinin bölgesi $V(p)$ ile gösterilmekte ve Voronoi hücresi olarak adlandırılmaktadır. Delaunay üçgenlemesi de Voronoi diyagramının grafiksel çifteşidir.

Voronoi diyagramının grafiksel çifteşinde her Voronoi hücresi için yani her mevzi için bir düğüm noktası vardır ve komşu iki düğüm arasında bir eğri mevcuttur. Bu $\mathcal{G}$ grafiğinin $Vor(P)$'nin her kenarı için bir eğriye sahip olduğu anlamına gelir. $Vor(P)$'nin köşe noktalarıyla $\mathcal{G}$'nin sınırlandırılmış yüzleri arasında birebir bir eşleşme mevcuttur.

$V(p)$ Voronoi hücresine karşılık gelen nokta p ve $V(q)$ Voronoi hücresine karşılık gelen nokta q olmak üzere, p ve q düğüm noktaları arasına $\overline{pq}$ doğru parçası yerleştirilsin. Bu işleme P'nin Delaunay Grafiği denir ve $D\mathcal{G}(P)$ ile gösterilir. Bir noktalar kümesinin Delaunay grafiğinin önemli özellikleri vardır. Bunlardan ilki bu grafiğin düzlemsel bir grafik olmasıdır; grafikte kesişen kenarlar yoktur.

Teorem 2.6: Bir noktalar kümesinin Delaunay grafiği düzlemsel bir grafiktir.

İspat: Bu teoremin ispatı için Voronoi diyagramının kenarlarının özelliklerinden faydalanılır. Teorem 2.3(ii)'de verilen özellik Delaunay grafiği için söylenirse,

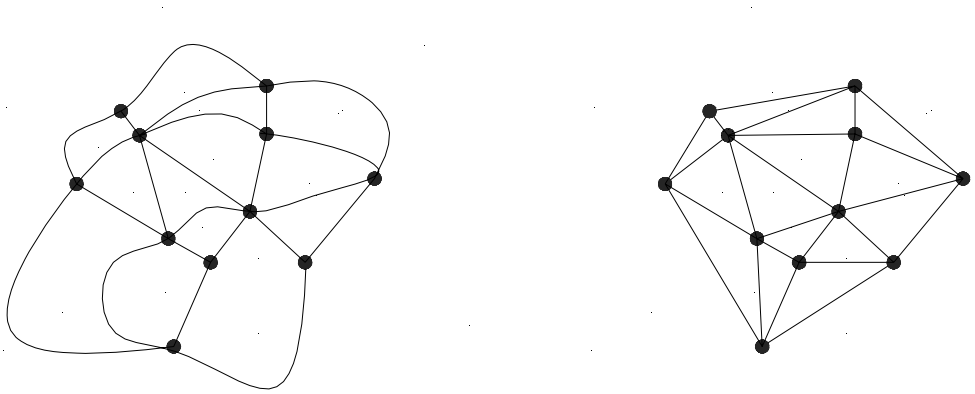
$\overline{p_ip_j}$ kenarı sadece ve sadece sınırında p_i ve p_j noktalarının olduğu ve içinde P'nin başka hiçbir mevzisini bulundurmayan bir C_{ij} çemberi mevcutsa $D\mathcal{G}(P)$ Delaunay grafiğinde yer alır.

Köşe noktaları p_i, p_j ve C_{ij}'nin merkez noktası olan bir t_{ij} üçgeni tanımlansın. t_{ij}'nin p_i'yi C_{ij}'nin merkezine bağlayan kenarı $V(p_i)$ Voronoi hücresi içinde ve p_j 'yi C_{ij}'nin merkezine bağlayan kenarı $V(p_j)$ Voronoi hücresi içinde yer almaktadır. $\overline{p_kp_l}$ kenarı da $D\mathcal{G}(P)$'nin diğer bir kenarı olsun ve t_{ij} ve C_{ij}'ye benzer şekilde t_{kl} ve ve C_{kl} tanımlansın.

$\overline{p_ip_j}$ ve $\overline{p_kp_l}$ kenarlarının kesiştiği kabul edilsin. p_k ve p_l C_{ij}'nin dolayısıyla t_{ij}'nin dışında yer almalıdır. Buna göre $\overline{p_kp_l}$ kenarı t_{ij}'nin C_{ij}'nin merkezine bağlı kenarlarından birini kesmelidir. Benzer şekilde, $\overline{p_ip_j}$'de t_{kl} üçgeninin C_{kl} çemberinin

merkezine bağlı kenarlarından birini kesmelidir. Buna göre, t_{ij}'nin C_{ij}'nin merkezine bağlı kenarlarından birinin t_{kl}'nin C_{kl}'nin merkezine bağlı kenarlarından birini kesmesi gerektiği sonucuna varılır. Ancak bu da bu kenarların ayrık Voronoi hücrelerinde olması gerekliliğine ters düşer.

P'nin Delaunay grafiği, Voronoi diyagramının grafiksel çifteşinin gömülmesiyle elde edilir (Şekil 2.8). Delaunay grafiği Voronoi diyagramının her köşe noktası için bir yüze sahiptir. Bir yüzü çevreleyen kenarlar o Voronoi köşesine bağlı Voronoi kenarlarına karşılık gelir. Özel olarak v köşesi $Vor(P)$'nin $p_1, p_2, \ldots, p_{\mathrm{k}}$ mevzilerinin Voronoi hücrelerinin köşesi ise, $DG(P)$'nin bu köşeye karşılık gelen yüzünün köşeleri $p_1, p_2, \ldots, p_{\mathrm{k}}$ olur. Öyleyse f yüzü hem k kenarlı bir çokgendir, hem de konveks bir yüzdür.



Şekil 2.8 Delaunay grafiği ve Delaunay üçgenlemesi

P'nin noktaları rastgele dağıtılmış ise dört noktanın aynı çember üzerinde bulunma olasılığı oldukça düşüktür. Eğer bir noktalar kümesi aynı çember üzerinde hiç dört nokta bulundurmuyorsa genel pozisyonundadır denir. Eğer P genel pozisyonunda ise Voronoi diyagramının bütün köşeleri 3. dereceye sahiptir ve bu yüzden $DG(P)$'nin bütün yüzleri üçgendir. Bu durum Delaunay grafiğine çoğu zaman Delaunay üçgenlemesi denmesinin sebebidir. Ancak Delaunay üçgenlemesini Delaunay grafiğine kenarlar eklenerek elde edilen bir üçgenleme olarak tanımlamak mümkündür.

$DG(P)$'nin bütün yüzleri konveks olduğundan böyle bir üçgenleme elde etmek oldukça basittir. Sadece P genel pozisyonunda ise ve buna bağlı olarak $DG(P)$ bir üçgenleme ise o zaman Delaunay üçgenlemesi tektir.

Voronoi diyagramı için geliştirilen teorem Delaunay grafiği için tekrarlanırsa;

Teorem 2.7: P düzlemde bir noktalar kümesi olsun.

(i) $p_i, p_j, p_k \in$ P noktaları sadece ve sadece üzerinde p_i, p_j ve p_k noktalarını barındıran çemberin içinde başka hiçbir nokta yoksa Delaunay grafiğinin aynı yüzünün köşeleridir.

(ii) $p_i, p_j \in$ P noktaları sadece ve sadece p_i ve p_j noktalarından geçen ve içinde P'nin diğer hiçbir noktasını içermeyen bir C çemberi varsa P'nin Delaunay grafiğinin bir kenarını oluşturur.

Bu teorem Delaunay üçgenlemelerinin aşağıdaki özelliğini ortaya koyar.

Teorem 2.8: P düzlemde bir noktalar kümesi ve T'de P'nin bir üçgenlemesi olsun. T üçgenlemesi eğer herhangi bir üçgenini çevreleyen çember içinde P'nin başka bir noktasını bulundurmuyorsa P'nin Delaunay üçgenlemesidir.

Üçgenleme bölümünde anlatıldığı gibi herhangi bir üçgenlemenin açı vektörü mümkün olduğunca büyükse yüzey modelleme ve yükseklik interpolasyonunda iyi sonuçlar vermektedir. Bunun için Delaunay üçgenlemelerinin açı vektörlerinin durumuna bakılacaktır.

Teorem 2.9: P düzlemde bir noktalar kümesi olsun. P'nin bir üçgenlemesi olan T sadece ve sadece P'nin Delaunay üçgenlemesi ise geçerli bir üçgenlemedir. Bu aynı zamanda herhangi bir Delaunay üçgenlemesinin geçerli bir üçgenleme olduğunu ortaya koyar.

Herhangi bir optimal açılı üçgenlemenin geçerli bir üçgenleme olması gerektiği göz önüne alınırsa P'nin herhangi bir optimal açılı üçgenlemesi P'nin Delaunay üçgenlemesidir. Eğer P genel pozisyonunda ise o zaman sadece bir geçerli üçgenleme vardır ve bu da optimal açılı üçgenleme olan tek Delaunay üçgenlemesidir. Eğer P genel pozisyonunda değilse Delaunay grafiğinin bütün üçgenlemeleri geçerlidir. Bütün bu Delaunay üçgenlemelerinin hepsi optimal açılı değildir, ancak bu üçgenlemelerin açı vektörleri çok küçük farklar gösterir. Genel pozisyonunda olmayan noktalar kümesinin üçgenlemelerinde minimum açı üçgenlemeden bağımsızdır. Dolaysıyla Delaunay grafiğini Delaunay üçgenlemesine çeviren bütün üçgenlemelerde minimum açı aynıdır. Aşağıdaki teoremde bu durum belirtilmiştir.

Teorem 2.10: P düzlemde bir noktalar kümesi olsun. P'nin optimal açılı bütün üçgenlemeleri P'nin Delaunay üçgenlemesidir. P'nin herhangi bir Delaunay üçgenlemesi minimum açıyı maksimize eder.

2.2.4 Delaunay üçgenlemesinin hesaplanması

Delaunay üçgenlemesi minimum açıyı maksimize ettiğinden örnek noktalar kümesinden çok yüzlü yüzey modelleri oluşturma gibi işlemlerde en iyi sonucu verir.

Voronoi diyagramı elde edildikten sonra Delaunay grafiği $DG(P)$ kolaylıkla elde edilebilir. Bundan sonra üçten fazla köşe noktasına sahip bütün yüzlerin üçgenlenmesiyle Delaunay üçgenlemesi elde edilir. Ancak doğrudan Delaunay üçgenlemesinin elde edilebileceği algoritmalar bulunmaktadır. Bu algoritmalar üçüncü bölümde tartışılacaktır.

2.3 Grafiksel Çifteşlik

Düzlemdeki bir noktanın x-koordinatı ve y-koordinatı olmak üzere iki parametresi vardır. Düzlemdeki bir doğrunun da eğimi ve y eksenini kestiği nokta olmak üzere yine iki parametresi vardır. Bu yüzden bir noktalar kümesi bir doğrular kümesine veya

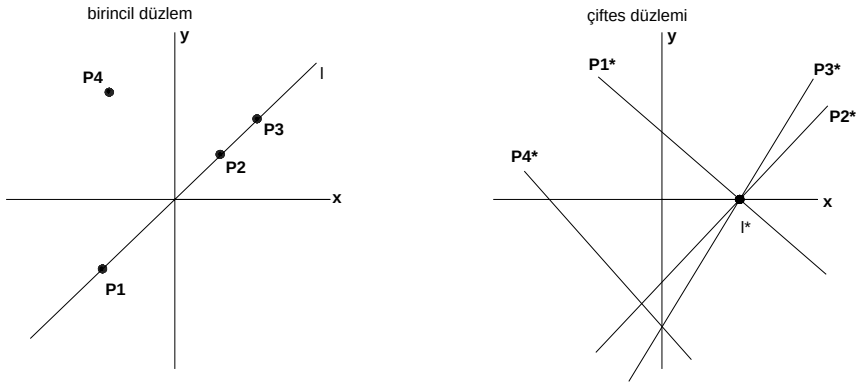
tersine bir doğrular kümesi bir noktalar kümesine haritalanabilir. Bu haritalama yapılırken noktalar kümesinin belli özellikleri, doğrular kümesinin belli özelliklerine transfer edilir. Örneğin bir doğru üzerindeki üç nokta, bir noktadan geçen üç doğru haline gelmektedir. Bu dönüşümü sağlayan çeşitli haritalama yöntemleri bulunmaktadır ve bu yöntemler çifteşlik dönüşümleri olarak adlandırılır. Bir nesnenin bir çifteşlik dönüşümü altındaki görüntüsüne nesnenin çifteşi denilmektedir. Temel bir çifteşlik dönüşümü şu şekilde verilebilir.

$p \coloneqq (p_x, p_y)$ düzlemde bir nokta olsun. p'nin çifteşi p^* ile gösterilen bir doğrudur ve

$p^* = (y = p_x x - p_y)$ ile tanımlanır.

$l^* = y = mx + b$ doğrusunun çifteşi ise $p^* = l$ olan p noktasıdır ve bu durumda

$l^* = (m, -b)$ olacaktır.



Şekil 2.9 Nokta ve doğrunun çifteş dönüşümü

Çifteşlik dönüşümü dikey doğrular için tanımlı değildir. Çoğu durumda dikey doğrular ayrı olarak ele alınabilir ve bir problem oluşturmaz. Diğer bir çözüm ise bulunulan sahnenin döndürülmesidir, böylelikle bu doğrular artık dikey değildir.

Çifteşlik dönüşümü nesneleri birincil düzlemden çifteş düzlemine aktarır. Birincil düzlemdeki belli temel özellikler çifteş düzleminde de geçerliliğini korur.

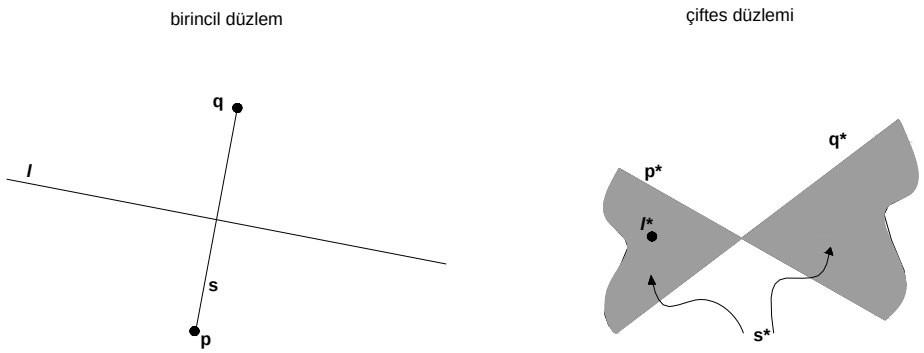
Ön Teorem 2.1 : p düzlemde bir nokta ve l'de düzlemde dikey olmayan bir doğru olsun. $o \longmapsto o^*$ çifteşlik dönüşümü aşağıdaki özelliklere sahiptir.

(i) Bu dönüşüm kesişimleri korur, $p \in l$ olması için gerek ve yeter şart $l^* \in p^*$ olmasıdır.

(ii) Bu dönüşüm sıralamaları da korur. p, sadece ve sadece l^*, düzlemde p^*'nin üzerinde yer alıyorsa l'nin üzerinde yer alır.

Şekil 2.9 bu özellikleri göstermektedir. p_1, p_2, p_3 noktaları birincil düzlemde l doğrusunda bulunmaktadır; çifteş düzlemindeki ${p_1}^*$, ${p_2}^*$, ${p_3}^*$ doğruları ise l^* noktasından geçmektedir. p_4 noktası ise birincil düzlemde l doğrusunun üzerinde yer almaktadır ve çifteş düzleminde de l^* noktası, ${p_4}^*$ doğrusunun üzerindedir.

Çifteş dönüşümleri noktalar ve doğrular dışında diğer nesnelere de uygulanabilir. Örneğin $s := \overline{pq}$ doğru parçasının çifteşi ne olacaktır? s^* için bir çözüm s'yi oluşturan bütün noktaların çifteşlerinin birleşimini bulmak olabilir, bu durumda elde edilecek olan sonsuz bir doğrular kümesidir. s'yi oluşturan bütün noktalar doğrusal olduğuna göre bu noktaların çifteşleri olan doğrular aynı noktadan geçer. Onların birleşimi, s'nin iki uç noktasının çifteşleri ile sınırlandırılmış çift takoz şekli meydana getirmektedir. s'nin uç noktalarının çifteşleri olan doğrular üst-alt ve sağ-sol olmak üzere iki tane çift takoz şekli tanımlamaktadır; s^* bunlardan sağ-sol olanıdır. Şekil 2.10 bir s doğru parçasının çifteşini göstermektedir. Bu şekilde aynı zamanda s'yi kesen bir l doğrusu da gösterilmiştir. l'nin çifteşi l^* noktası da s^* içinde yer almaktadır.

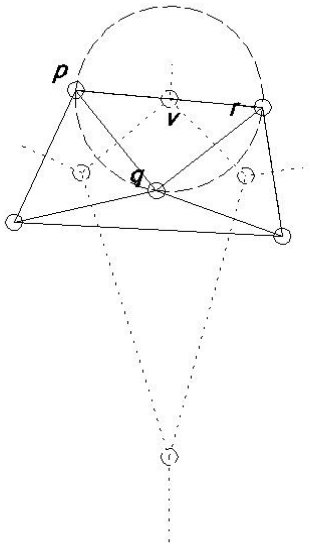


Şekil 2.10 Doğru parçasının çifteş dönüşümü

Çifteşlik dönüşümlerinin kullanışlı olmasının temel sebebi karşılaşılan bir problemde probleme farklı bir perspektiften bakma imkanı sağlamasıdır. Eğer bir problem çifteş düzleminde çözülebiliyorsa, bu çözüm tekrar birincil düzleme haritalanarak birincil düzlemde de çözüm elde edilmiş olacaktır. Aslında her iki düzlemde de problem aynıdır ancak aynı probleme farklı bir açıdan bakmak çözüme ulaşmayı kolaylaştıran bir yol olabilmektedir.

2.3.1 Voronoi – Delaunay çifteşliği

Voronoi kenarlarının kesişim noktaları olan Voronoi diyagramının köşeleri Delaunay üçgenlemesinin gerçekleştirilmesinde çok önemli bir rol oynar. Şekil 2.11'de gösterildiği gibi bir Voronoi köşesinin en büyük boş çemberi çizildiğinde pqr noktalarını içerir ve bu da Delaunay üçgenlemesinin temelini oluşturur.



Şekil 2.11 Voronoi-Delaunay çifeşliğinin gösterimi

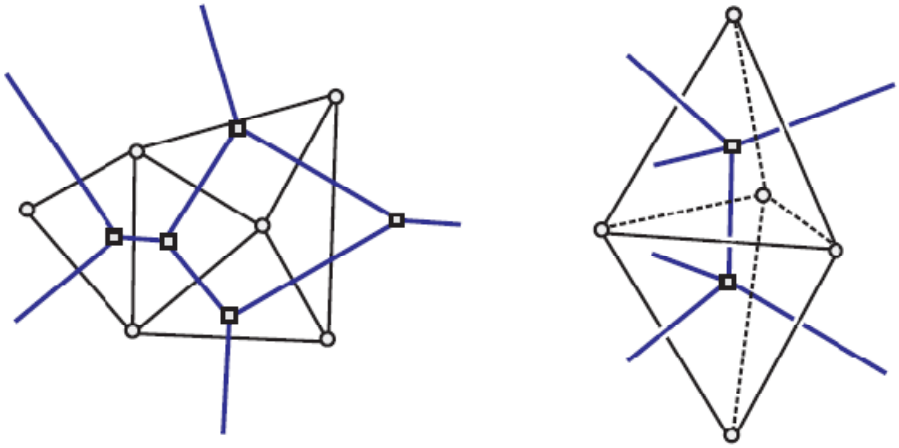
Çifteşlik tanımından yola çıkılırsa, bir $\mathcal{G}$ grafiğinin köşeleri, $\mathcal{G}$'nin çifteşi $\mathcal{G}^*$'de bir yüze karşılık gelir. Bu yüzün de köşeleri $\mathcal{G}$ grafiğinin yüzleridir. Bunun yanında $\mathcal{G}^*$ çifteş grafiğindeki iki köşe noktası eğer $\mathcal{G}$ grafiğinde bu köşe noktalarına karşılık gelen yüzler ortak bir kenarı paylaşıyorlarsa bir kenar ile birbirine bağlıdır. Buradaki durum için daha da özele inilirse,

Her $p \in P$ köşe noktası bir $V(p)$ Voronoi hücresine karşılık gelir.

Her $t \in Del(P)$ üçgeni $Vor(P)$'nin bir köşe noktasına karşılık gelir.

Her $e(p,q) \in Del(P)$ kenarı $V(p)$ ve $V(q)$ Voronoi hücreleri arasındaki sınır kenara karşılık gelir.

Özel olarak söylenirse, bir P noktalar kümesinin Voronoi diyagramı $Vor(P)$ ve Delaunay üçgenlemesi $Del(P)$ birbirlerinin grafiksel çifteşidir. Şekil 2.12'de 2 ve 3 boyutlu düzlemlerde aynı noktalar kümelerinin Voronoi diyagramı ve Delaunay üçgenlemesi gösterilmektedir.



Şekil 2.12 İki ve Üç-boyutlu uzaylarda Voronoi – Delaunay çifteşliği

3. ALGORİTMALAR ve GENEL UYGULAMALAR

3.1 Algoritmalar

3.1.1 Giriş

Bu bölümde Voronoi diyagramının ve Delaunay üçgenlemesinin elde edilmesi için kullanılan çeşitli algoritmalar ortaya konacaktır. İncelemenin basitliği için P noktalar kümesindeki n noktanın herhangi dördünün aynı çember üzerinde bulunmadıkları ve herhangi üçünün doğrusal olmadıkları kabul edilecektir. Bu durumda $DG(P)$ doğrudan $Del(P)$ olacaktır. Ancak verilen algoritmalar genel pozisyon yaklaşımı olmadan da çalışacaktır. Bunun yanında algoritmalar Öklit uzayından metrik uzaylara ve noktalardan mevzilere genelleştirilebilir.

Voronoi grafikleri gibi düzlemsel grafiklerde çalışmak üzere tasarlanmış veri yapıları "$DCEL$" (Doubly Connected Edge List) (Muller ve Preparata, 1978) ve "$Quad\ Edge$" yapılarıdır (Guibas ve Stolfi, 1985). Her iki yapıda her e kenarı için şu bilgileri saklayan kayıtlar tutulur: e'nin iki uç noktasının isimleri, bitiş noktalarındaki diğer kenarlar için saat yönünde veya saat yönünün tersi yönünde bilgisi ve e'nin her iki tarafındaki yüzlerin kenarları. He iki yapıda da $O(n)$ bellek kapasitesi kullanılır.

Voronoi diyagramları ile Delaunay üçgenlemeleri grafiksel çifteş olduklarından bir noktalar kümesi için Voronoi diyagramının hesaplanması Delaunay üçgenlemesinin lineer zamanda hesaplanabilmesi için yeterlidir. Çizelge 3.1'de çeşitli algoritmaların en kötü durum çalışma sürelerinin bir karşılaştırılması verilmektedir.

Çizelge 3.1 Bazı algoritmaların en kötü durum çalışma sürelerinin karşılaştırılması (Fortune 2004)

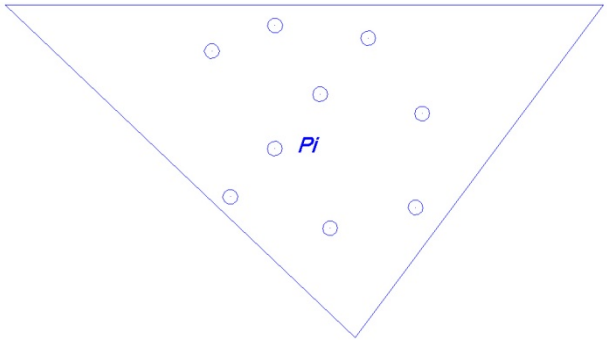
Algoritma	Boyut	En kötü Durum
Artırımlı ekleme	2	$O(n\log n)$
Artırımlı ekleme	≥ 3	$O(n^{\lceil \frac{d}{2} \rceil})$
R^{d+1}	2	$O(n\log n)$
Fortune (tarama çizgisi)	2	$O(n\log n)$
Böl ve Fethet	2	$O(n\log n)$

Burada verilecek ilk iki algoritma doğrudan Delaunay üçgenlemesini hesaplayan ve artırımlı ekleme algoritmaları sınıfına giren algoritmalardır. Artırımlı ekleme algoritmaları üçgenlenecek noktaları birer birer ekleyerek çalışır (Green ve Sibson, 1978). Bir nokta eklendikten sonra Delaunay kriteri tekrar kontrol edilir. Genellikle bu algoritmalarda bütün noktaları kapsayacak süper üçgen olarak adlandırılan bir üçgenle algoritma başlatılır (Su ve Drysdale, 1995).

3.1.2 Bowyer – Watson algoritması

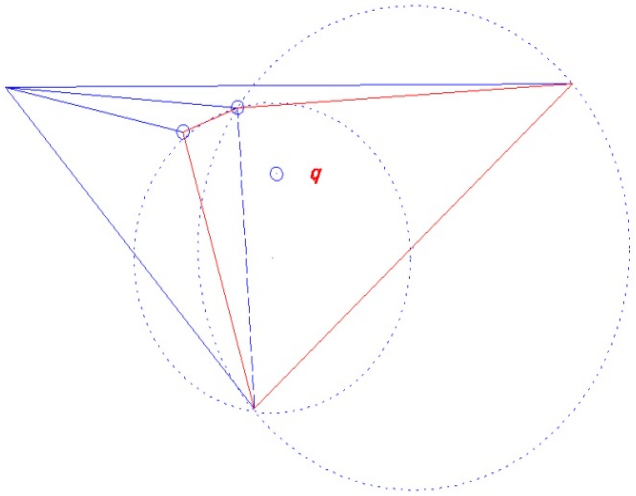
Bowyer – Watson algoritması herhangi bir boyut sayısında, sonlu sayıdaki noktadan oluşan bir nokta kümesinin Delaunay üçgenlemesinin hesaplanması için kullanılan bir yöntemdir. Bu algoritma artırımlı ekleme algoritmaları sınıfında incelenebilir.

Algoritma bazen sadece Bowyer algoritması ya da sadece Watson algoritması olarak adlandırılır. Adrian Bowyer (1981) ve David Watson (1981) aynı zamanda birbirlerinden bağımsız olarak bu algoritmayı duyurmuş ve her ikisi de "Computer Journal" dergisinin aynı sayısında birer makale yayınlamışlardır.

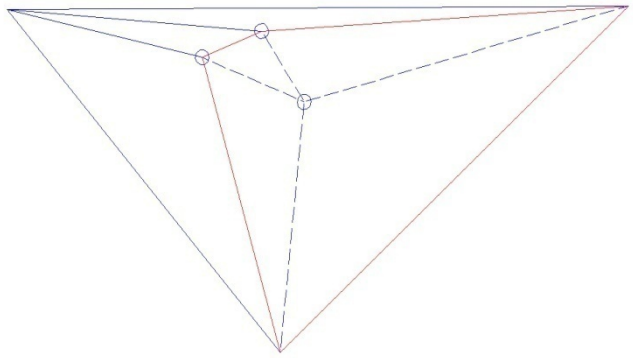


Şekil 3.1 Bütün noktaları içeren süper üçgen

Eğer bir başlangıç üçgeni verilmemiş ise, P noktalar kümesindeki bütün noktaları kapsayacak süper üçgen oluşturulur (Şekil 3.1). Algoritma P noktalar kümesinin tüm noktalarının birer birer üçgenlemeye eklenmesiyle çalışır. Her ekleme işleminde hangi üçgenlerin çevreleyen çemberlerinin yeni eklenen noktayı içerdiğini belirlemek üzere arama işlemi yapılır (Şekil 3.2). Sonra bu üçgenler silinir ve böylelikle yeni eklenen p noktasını içeren poligon oluşur. Bu poligona ekleme poligonu denir. Sonra p noktası ile poligonun köşeleri arasına kenarlar çizilerek yeni üçgenleme oluşturulur (Şekil 3.3). P noktalar kümesindeki bütün noktalar kümesi eklendikten sonra Delaunay üçgenlemesi $Del(P)$ elde edilmiş olur. Son olarak yapılması gereken süper üçgeni ve ona bağlı bütün kenarları üçgenlemeden çıkarmaktır (Zimmer 2005).



Şekil 3.2 Bowyer-Watson algoritmasında üçgenlemeye yeni bir noktanın eklenmesi

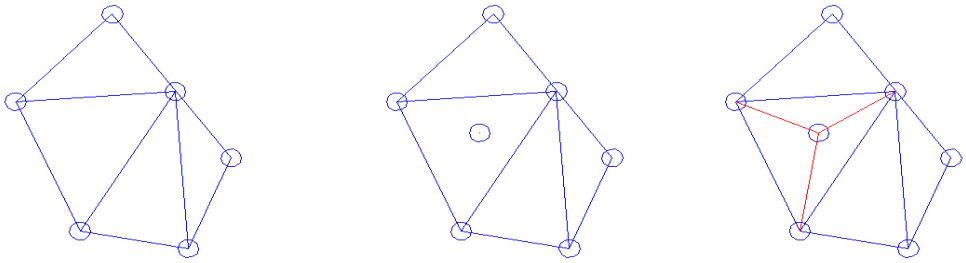


Şekil 3.3 Eklenen nokta sonrasında kenarların güncelleştirilmesi

Bowyer – Watson algoritması en yalın halinde kötü veya bozuk bir yüzey örgüsü oluşturabilir. Eklenen noktanın hangi çevreleyen çemberler tarafından içerildiğinin bulunması aşamasında algoritma eklemeden etkilenen bazı üçgenleri silmekte başarısız olabilir. Bu durum çemberlerin hesaplanmasındaki yuvarlama hatasından kaynaklanır. Böyle bir durumda boş olması gereken poligonun içinde bir ya da daha fazla üçgen kalacaktır ve poligonun yeniden üçgenlemesi aşamasında hatalı bir yüzey oluşacaktır. Bowyer – Watson algoritması ekleme poligonunun bulunmasında süper üçgenden derinlik öncelikli arama yapısı kullanılarak kararlı hale getirilebilir.

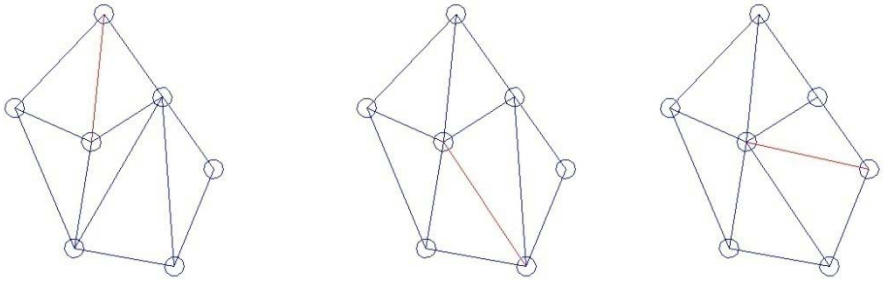
3.1.3 Lawson algoritması

Lawson algoritması da artırımlı ekleme algoritmaları sınıfına girmektedir (Lawson 1977). Bowyer – Watson algoritmasından farklı olarak Lawson algoritması çevreleyen çemberleri bulup üçgenleri silerek ekleme poligonu oluşturmak yerine aynı sonucu elde etmek için kenar çevirme işlemini kullanılır. Böylelikle muhtemel bir hatalı yüzey oluşması da engellenmiş olacaktır.



Şekil 3.4 Lawson algoritmasında üçgenlemeye yeni bir noktanın eklenmesi

Bu algoritmada da eğer bir başlangıç üçgeni verilmemişse bütün noktaları içeren süper üçgen kullanılır. Her adımda P noktalar kümesinden bir nokta eklenir. Her ekleme işleminden sonra yeni nokta ile bu noktayı içeren üçgenin köşe noktaları arasına kenarlar oluşturularak üçgenleme yapılır (Şekil 3.4). Sonra yeni oluşan üçgenler için Delaunay kriteri kontrol edilir. Bunun için oluşan üçgenlerin çevreleyen çemberleri içinde başka nokta bulunup bulunmadığına bakılır. Eğer böyle bir durum varsa kenar çevirme işlemi uygulanır (Şekil 3.5). Algoritma sonraki noktanın eklenmesiyle devam eder. Son aşamada yine süper üçgen ve ona bağlı kenarlar silinir (Zimmer 2005).



Şekil 3.5 Lawson algoritmasında geçersiz kenarların kenar çevirme işlemine tabi tutulması

3.1.3.1 Algoritmanın çalışması

Lawson algoritması rastgele artırımlı bir algoritmadır. Buna göre algoritma üçgenlemeye rastgele sırada noktalar ekler ve o anki noktaların Delaunay üçgenlemesini oluşturur. p_r noktasının o anki üçgenlemeye eklendiği düşünülsün. Öncelikle bu noktayı hangi üçgenin içerdiği belirlenir ve bu noktadan üçgenin köşe noktalarına kenarlar eklenir. Eğer p_r bir e kenarı üzerine düşerse o zaman bu noktadan e kenarını paylaşan üçgenlerin diğer köşelerine kenarlar eklenmelidir. Şekil 3.5'de bu durum gösterilmektedir. Böylelikle yeni bir üçgenleme elde edilmiş olur. Ancak bu üçgenleme Delaunay üçgenlemesi olmayabilir. Bunun sebebi, hem p_r'nin eklenmesiyle oluşan kenarların hem de bu ekleme işleminden etkilenen kenarların geçersiz kenarlar olabileceğidir. Bu durumu düzeltmek için bütün şüpheli kenarlar için KENARI_GEÇERLİLEŞTİR yordamı çağrılır.

```
Algoritma DELAUNAY_ÜÇGENLEMESİ(P)

Girdi: Düzlemde n+1 noktadan oluşan P noktalar kümesi

Çıktı: P'nin Delaunay üçgenlemesi
```

p_0 P'nin en büyük y koordinatına sahip en üstteki noktası olsun.

p_{-1} ve p_{-2} P kümesinin bütün noktalarının $p_0p_{-1}p_{-2}$ üçgeninde içerileceği şekilde R^2 uzayında göreceli olarak çok uzak iki nokta olsun.

T üçgenlemesini $p_0p_{-1}p_{-2}$ üçgeninden oluşan tek bir üçgenle başlat.

P'nin $\{p_1, p_2, \ldots\ldots\ldots, p_n\}$ noktalarının rastgele bir permütasyonunu hesapla.

for(r: 1' den n'e kadar)

do(*p_r'yi T'ye dahil et*)

p_r'yi içeren $p_ip_jp_k \in T$ üçgenini bul.

if (p_r $p_ip_jp_k$ üçgeninin içinde ise)

then

(p_r noktasından $p_ip_jp_k$ üçgeninin üç köşe noktasına $p_ip_jp_k$ üçgenini üç üçgene bölecek şekilde kenarlar ekle.)

KENARI_GEÇERLİLEŞTİR$(p_r, \overline{p_ip_j}, T)$

KENARI_GEÇERLİLEŞTİR$(p_r, \overline{p_jp_k}, T)$

KENARI_GEÇERLİLEŞTİR$(p_r, \overline{p_kp_i}, T)$

else(*$p_rp_ip_jp_k$ 'nın $\overline{p_ip_j}$ kenarı üzerinde ise*)

p_r noktasından p_k'ya ve $\overline{p_ip_j}$ kenarını paylaşan diğer üçgenin üçüncü köşesine iki üçgeni dört üçgene bölecek şekilde kenarlar ekle.

KENARI_GEÇERLİLEŞTİR$(p_r, \overline{p_ip_l}, T)$

KENARI_GEÇERLİLEŞTİR$(p_r, \overline{p_lp_j}, T)$

KENARI_GEÇERLİLEŞTİR($p_r, \overline{p_j p_k}, T$)

KENARI_GEÇERLİLEŞTİR($p_r, \overline{p_k p_i}, T$)

p_{-1} ve p_{-2} noktalarını ve onlara bağlı bütün kenarları T'den çıkar.

return T

Bu algoritma içinde nokta eklenip yeni kenarlar oluşturulduktan sonra oluşan üçgenlemenin Delaunay üçgenlemesine çevrilmesi gereklidir. Teorem 2.9'da belirtildiği gibi eğer bir üçgenlemenin bütün kenarları geçerli kenarlar ise Delaunay üçgenlemesidir. O yüzden algoritma içinde geçerli bir üçgenleme elde edilene kadar geçersiz kenarlar kenar çevirme işlemine tabi tutulur. Buradaki sorun p_r eklendiğinde hangi kenarların geçersiz duruma gelebileceğidir. Daha önceden geçerli bir $\overline{p_i p_j}$ kenarı ancak kendine komşu iki üçgende bir değişme olduğunda geçersizleşebilir. Buna göre sadece yeni oluşan üçgenlerin kenarları kontrol edilmelidir. Bu işlem KENARI_GEÇERLİLEŞTİR yordamı yardımıyla gerçekleştirilir. Bu yordam kenarları test edip gerekli ise çevirir. Eğer bir kenar çevirme işlemi uygulanırsa, üçgenin diğer kenarları da geçersiz duruma gelebilir. Bu yüzden yordam yinelemeli olarak kendini çağırmaktadır (de Berg vd. 2008).

KENARI_GEÇERLİLEŞTİR($p_r, \overline{p_i p_j}, T$)

(*p_r dahil edilen nokta, $\overline{p_i p_j}'$de geçersiz olma ihtimali olan kenardır*)

if($\overline{p_i p_j}$ kenarı geçersiz ise)

then($p_i p_j p_k$ $\overline{p_i p_j}$ boyunca $p_r p_i p_j'$ye komşu olan üçgendir.)

(*$\overline{p_i p_j}'$yi çevir*) $\overline{p_i p_j}'$yi $\overline{p_r p_k}$ ile değiştir.

KENARI_GEÇERLİLEŞTİR($p_r, \overline{p_i p_k}, T$)

KENARI_GEÇERLİLEŞTİR($p_r, \overline{p_k p_j}, T$)

3.1.4 R^{d+1} Projeksiyon algoritması

Bu algoritma Delaunay üçgenlemesinin hesaplanması problemini konveks zarf hesaplanması problemine dönüştürür. Algoritmanın ismi bir üst boyuta projeksiyon yapılmasından kaynaklanır. Algoritmanın özetle yaptığı iş bütün noktaları bir boyut yukarıda, orijine merkezlenmiş bir paraboloide yansıtmaktır. Bir üst boyutta konveks zarf hesaplanır ve hesaplanan konveks zarfın alt kısmı eski boyuta tekrar yansıtılır. Bu da Delaunay üçgenlemesini verir.

Bu durumda önce konveks zarfın tanımını yapmak doğru olacaktır. n boyutunda P noktalar kümesinin konveks zarfı P'yi içeren bütün konveks kümelerinin kesişimidir (http://www.wolfram.mathworld.com 2008).

İki boyutlu Delaunay üçgenlemesi $Del(P)$ üç boyutlu konveks zarfın hesaplamasıyla elde edilebilir. Üçgenlemesi yapılacak bir P noktalar kümesi olsun. Öncelikle bir boyut yukarıdaki paraboloidin -ki burada 3 boyutlu olacaktır- tanımlanması uygun olacaktır. Üç boyuttaki bu paraboloit $f(x,y) = (x, y, x^2 + y^2)$ ile tanımlanır ve Ω ile gösterilir. 2 boyuttaki P noktalar kümesinin noktalarının Ω üzerindeki yansımaları $\hat{p}$ ile gösterilsin. Bu durumda $\hat{p} = (p_x, p_y, p_x^2 + p_y^2)$ olacaktır. Yansıtılan $\hat{p}$ noktalarındaki tanjant düzlemleri de $\hat{p}^*$ ile ifade edilsin. İki boyuttaki tanjant doğrularının üst boyuttaki eşdeğeri olan tanjant düzlemleri şu şekilde tanımlanabilir.

(x_0, y_0) $z = f(x,y)$ yüzey fonksiyonunda herhangi bir nokta olsun. Bu yüzey (x_0, y_0) noktasında $z = f(x,y) + f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$ ile verilen ve dikey olmayan bir tanjant düzlemine sahiptir (http://www.wolfram.mathworld.com 2008).

Buradaki problem için bu ifade uygulanırsa $\hat{p}^*$ tanjant düzlemleri

$$z = 2p_x x + 2p_y y - p_x^2 - p_y^2 \qquad (3.1)$$

şeklinde elde edilir.

Herhangi üç $p, q, r \in P$ noktaları için $V(p)$, $V(q)$, $V(r)$ Voronoi hücreleri bir v Voronoi köşesini paylaşırlar. Tanjant düzlemleri $\hat{p}^*, \hat{q}^*, \hat{r}^*$ $\hat{p}, \hat{q}, \hat{r}$ noktalarından geçmektedir. Bu tanjant düzlemleri v üzerindeki uzayda v^* noktasında kesişir. Tanjant düzlemlerinin doğası gereği bu v^* noktası, v'nin Ω üzerindeki yansıması $\hat{v}$'nün altındadır.

Şimdi de bu v^* noktasında durulduğu ve paraboloide bakıldığı düşünülsün. Dış yüzeyde yani sınırda görülen noktalar, tanjant düzlemleri üzerinde bulunulan v^* noktasından geçen noktaların sınırlarıdır. Ω üzerindeki bu sınır Γ ile gösterilir. Γ üzerindeki noktalar $\hat{g}$ ve tanjant düzlemleri $\hat{g}^*$ ile gösterilsin. Γ'nin 2 boyuttaki düzleme yansımaları v noktasına eşit uzaklıktaki g noktalarını verecektir. g noktaları da merkezi v noktası olan bir çember teşkil eder. v noktası $V(p), V(q), V(r)$ Voronoi hücrelerinin ortak köşe noktası olduğundan bu çemberde pqr Delaunay üçgeninin çevreleyen çemberi olacaktır.

Peki, bu durum nasıl konveks zarfa karşılık gelmektedir? p, q ve r noktaları tek bir çember tanımlamakta olduklarından Γ, paraboloidin $\hat{p}, \hat{q}, \hat{r}$ noktalarından geçen düzlemle kesişimidir. p, q, r noktaları, bu noktaların Voronoi hücreleri bir Voronoi köşesini paylaştıklarından bir Delaunay üçgeni oluşturdukları bilinmektedir. Dolayısıyla bu çevreleyen çemberin P'nin diğer bütün noktalarıyla ilgisi yoktur. Bu da diğer bütün noktaların paraboloit üzerindeki yansımalarının Γ'den geçen düzlemin üzerinde yer alması anlamına gelmektedir. Dolayısıyla Ω üzerinde bu düzlemin altında kalan kısım yansıtılan noktaların hiçbiri etkilenmeden kesilebilir. Her kesme işlemi paraboloidin bir parçasını yok eder ve $\hat{p}, \hat{q}, \hat{r}$'den geçen bir üçgen bırakır. $Del(P)$'nin bütün p, q, r noktaları için bu üçgen yüzleri yansıtılan noktaların konveks zarfını oluşturur. Dolayısıyla sonuç şöyle söylenebilir. Konveks zarf yüzleri ve onun sınırları, Ω üzerine yansıtılmış Delaunay üçgenleridir (Zimmer 2005).

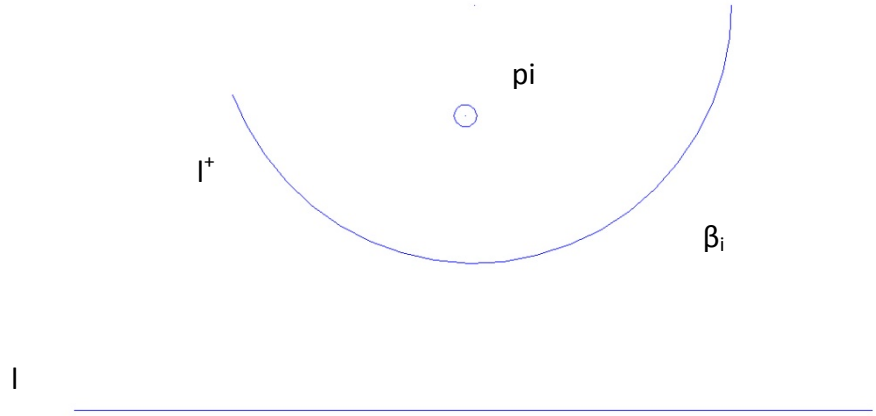
Buna göre Delaunay üçgenlemesinin hesaplanması konveks zarf hesaplanması ve iki projeksiyon işleminden oluşacak hale getirilmiştir. Herhangi bir kümenin konveks

zarfının hesaplanması için çok çeşitli algoritmalar geliştirilmiştir. Qhull algoritmasının karmaşıklığı $O(n\log v)$ olarak verilmektedir (http://www.qhull.com 1995). n ve v sırasıyla giriş ve çıkış noktalarının sayısıdır. Buradaki giriş ve çıkış noktalarının sayısı aynı olduğundan karmaşıklık $O(n\log n)$ olarak verilebilir. Yansıtma işlemleri lineer sürede yapılabildiğine göre R^{d+1} projeksiyon algoritmasının toplam karmaşıklığı $O(n\log n)$ olacaktır.

3.1.5 Fortune algoritması

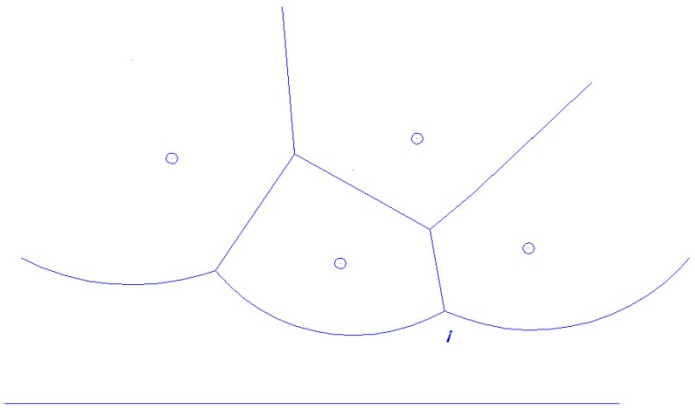
Fortune algoritması düzlemdeki bir nokta kümesinden Voronoi diyagramının üretilmesinde kullanılan bir düzlem tarama algoritmasıdır. Bu algoritma $O(nlogn)$ süre ve $O(n)$ bellek kapasitesi kullanır. Algoritma ilk olarak Steven Fortune tarafından "A sweepline algorithm for Voronoi diagrams" adı ile yayınlanmıştır (Fortune 1987).

Algoritma bir tarama çizgisi ve bir kıyı çizgisinden oluşur ki bu çizgilerin her ikisi de algoritmanın işlemesi sırasında düzlem boyunca hareket eder (Şekil 3.6). Tarama çizgisi düz bir çizgidir ve klasik olarak düzlemde soldan sağa doğru ya da yukarıdan aşağıya doğru hareket eden bir çizgi olarak düşünülebilir. Algoritmanın çalışması sırasında herhangi bir anda tarama çizgisinin üstündeki mevziler Voronoi diyagramına entegre edilmiştir, altındaki mevziler ise henüz işlenmemiştir.



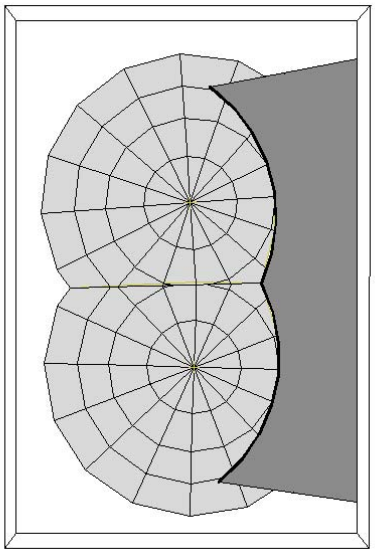
Şekil 3.6 Tarama çizgisi ve kıyı çizgisi

Kıyı çizgisi ise aslında bir çizgi değildir, tarama çizgisinin üstünde, birçok parabolden oluşan karmaşık bir eğridir (Şekil 3.7). Bu eğri tarama çizgisinin altındaki çizgileri göz önüne almadan düzlemin Voronoi diyagramı hesaplanan bir parçasını içine alır. Tarama çizgisinin üstündeki her nokta için, o noktadan ve tarama çizgisinden eşit uzaklıktaki noktalardan oluşan bir parabol tanımlanabilir ve kıyı çizgisi bu parabollerin birleşiminin sınırıdır.



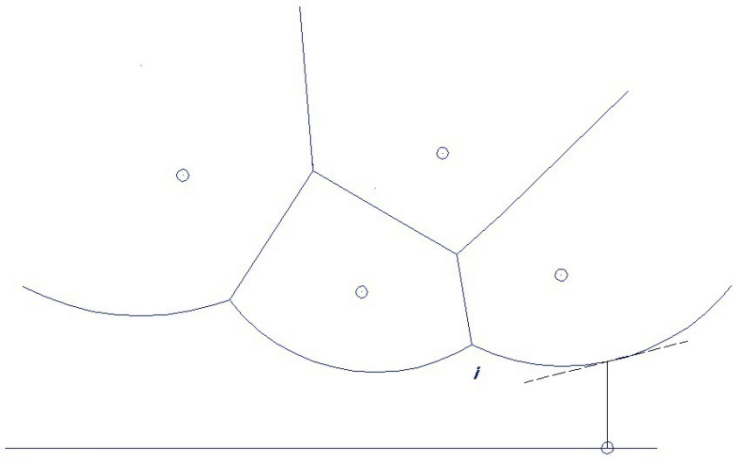
Şekil 3.7 Tarama çizgisinin ilerlemesi

Voronoi diyagramının öğelerinin oluşmasında iki tip olay vardır. Bir "nokta olayı" bir Voronoi kenarının oluşmasını sağlar. Bir "çember olayı" ise Voronoi kenarının oluşmasının yanında Voronoi köşesinin de oluşmasını sağlamaktadır.



Şekil 3.8 Tarama çizgisi ilerlerken d+1 boyuttaki görüntü

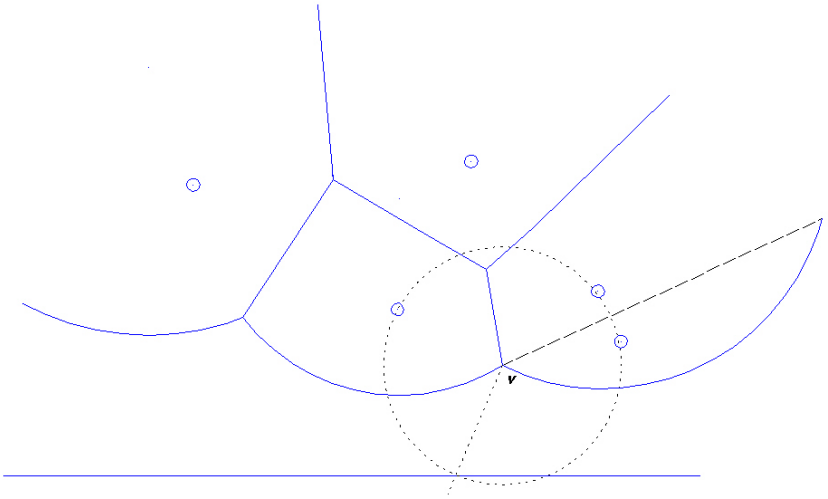
Düzlemde bir P noktalar kümesi verildiğinde düzlemi yukarıdan aşağıya doğru tarayacak bir l doğrusu düşünülsün. Bu tarama çizgisi olacaktır. l doğrusunun üzerinde kalan yarı düzlem l^+ olarak adlandırılsın. Bir $p_i \in P$ noktası için β_i'de odağında p_i'nin olduğu yatay parabol olsun. β_i, l^+'da p_i'den ve tarama çizgisi l'den eşit uzaklıktaki noktalardan oluşur. Kıyı çizgisi ise l^+ yarı düzlemindeki noktaların parabollerinin alt sınırıdır. Tarama çizgisi aşağıya doğru hareket ettikçe kıyı çizgisi de doğal olarak onu takip eder ve kıyı çizgisinin yapısı gereği aşağı doğru ilerledikçe giderek düzleşir. Voronoi kenarları bir nokta olayının gerçekleşmesi ile ortaya çıkar. Tarama çizgisi yeni bir p_n noktasına ulaştığında bir nokta olayı gerçekleşir (Şekil 3.9). Önce p_n noktasından kıyı çizgisine doğru dikey bir çizgi çizilir. Bu da yeni bir Voronoi kenarının doğmasını sağlar. Bu dikey çizgi yeni bir β_n parabolüne genişler ve kıyı çizgisinde bir boşluk oluşturur. β_n ile kıyı çizgisinin kalan kısmının kesişimleri yeni Voronoi kenarını çizmeye başlar.



Şekil 3.9 Nokta olayının gerçekleşmesi

Voronoi köşelerinin oluşması için ise çember olayının gerçekleşmesi gerekmektedir. Çember olayı iki Voronoi kenarının bir kesişim noktasında buluştuğu anda oluşur. Diğer bir deyişle kıyı çizgisinin sıralı üç yayı için kenardaki iki yayın ortadaki üzerine büyümesi esnasında gerçekleşir (Şekil 3.10). Sıralı yay üçlemesi p_i, p_j, p_k noktaları için oluşan paraboller $\beta_i, \beta_j, \beta_k$ olmak üzere kıyı çizgisinin bir bölümüdür. β_i ve β_k'nın kesişim noktası bir Voronoi köşesinin oluşmasını sağlar. Aynı zamanda bu olayda yeni bir Voronoi kenarı da oluşur. Çünkü bu anda β_i ile β_k arasında yeni bir kesişim çizgisi izlenmeye başlanmıştır. Bir çember olayı ile oluşturulan Voronoi kenarı tam olarak parabolleri sıralı yay üçlemesini oluşturan p_i, p_j, p_k noktalarının tanımladığı çemberin merkez noktasıdır. Bu yüzden bu olaya çember olayı adı verilmektedir.

Bu olaylar algoritmanın yapıtaşları olduklarından ne zaman oluştuklarının izlenmesi gerekmektedir. Nokta olayının gerçekleşmesinin belirlenmesi kolaydır, tek yapılması gereken tarama çizgisinin yeni bir noktaya ulaştığının kontrol edilmesidir. Çember olayının gerçekleşmesinin belirlenmesi için kıyı çizgisindeki sıralı yay üçlemelerinin izlenmesi gereklidir. Herhangi bir sıralı yay üçlemesi ikili haline geldiğinde çember olayı gerçekleşir. Tarama çizgisi bütün noktaları taradığında ve sıralı yay üçleme listesi boşaldığında geriye kalanlar Voronoi kenarları ve köşeleri olacaktır. Bu da P noktalar kümesinin Voronoi diyagramı $Vor(P)$'yi verir (de Berg vd. 2008).



Şekil 3.10 Çember olayının gerçekleşmesi

3.1.5.1 Algoritmanın çalışması

Buraya kadar Fortune algoritması özetlenmiştir. Artık algoritmanın nasıl çalışması gerektiği bilindiğine göre Voronoi diyagramının oluşturulabilmesi için gerekli veri yapılarının belirlenmesi gerekmektedir. Birinci veri yapısı o ana kadar hesaplanan Voronoi diyagramının tutulması için gerekli olandır. Bunun yanında olay kuyruğunu tutan ve tarama çizgisinin durumunu tutan iki standart veri yapısına ihtiyaç vardır. Buradaki ikinci yapı aynı zamanda kıyı çizgisini de teşkil etmektedir.

Voronoi diyagramının oluşturulan kısmını tutmak için klasik veri yapısı olan $DCEL$ (*doubly-connected edge list*) uygun olacaktır. Ancak Voronoi diyagramı tam anlamıyla bir alt grafik olmadığından, yani yarı doğru ve doğrular olabileceğinden bu bölümler bu veri yapısında tutulamaz. Bu durum diyagramın oluşturulması aşamasında bir problem oluşturmaz. Çünkü kıyı çizgisinin tutulduğu yapı, $DCEL$'nin ilgili kısmına ulaşmayı mümkün kılar. Hesaplama bittiğinde ise geçerli bir $DCEL$'nin oluşturulması gerekir. Bu son aşamada sahneye bütün Voronoi köşeleri içerecek kadar büyük bir kutu eklenir. Sonuçta elde edilen, bir kutu ile çevrelenmiş Voronoi diyagramı olacaktır.

Kıyı çizgisi ise T ile gösterilen bir dengeli ikili arama ağacı yapısı ile ifade edilir. Bu durum yapısını oluşturacaktır. Bu yapının yaprakları kıyı çizgisinin yaylarına karşılık gelecektir. Bu dengeli yapı olduğundan sıralı olacaktır. Yani en soldaki yaprak kıyı çizgisinin en solundaki yayı ifade edecektir. Her μ yaprağı ilgili yayı ifade eden mevziyi tutacaktır. T'nin iç düğüm noktaları ise Kıyı çizgisindeki kırılma noktalarını ifade eder. Bir kırılma noktası bir iç düğümde $< p_i, p_j >$ şeklinde bir veri grubu ile tutulur. Burada p_i kırılma noktasının solundaki ve p_j de sağındaki parabolü ifade etmektedir. Kıyı çizgisinin bu veri yapısı kullanılarak $O(\log n)$ sürede yeni bir mevzinin üzerinde yer alan kıyı çizgisinin yayı bulunabilir. İç düğümde ise basit olarak yeni mevzinin x-koordinatıyla kırılma noktasının x-koordinatı karşılaştırılır. Bu koordinatlar da, tarama çizgisinin pozisyonu ile mevzilerin veri grubundan sabit sürede elde edilebilir. Burada parabollerin tutulmadığına dikkat edilmelidir.

T içinde aynı zamanda tarama işlemi sırasında kullanılan diğer iki veri yapısına işaretçiler bulunmaktadır. T'nin α yayını ifade eden her yaprağı olay kuyruğunda α'nın yok olacağı çember olayını ifade eden düğüme, bir işaretçiye sahiptir. Eğer α'nın yok olacağı bir çember olayı yoksa veya bu olay henüz belirlenmemişse bu işaretçi $null$ değerine sahip olacaktır. Son olarak her iç düğümünde Voronoi diyagramının $DCEL$'indeki yarı kenarlara bir işaretçi vardır. Daha özel olarak söylenirse v iç düğümü, v ile ifade edilen kırılma noktası ile oluşturulacak kenarın yarı kenarlarına bir işaretçiye sahiptir.

Olay kuyruğu Q ise bir öncelik kuyruğu olarak geliştirilmektedir. Burada öncelik olayların y-koordinatı olacaktır. Bu yapı şu an bilinen gelişecek olayların kaydını tutmaktadır. Bir nokta olayı için basitçe noktanın kendisi tutulur. Bir çember olayı için ise tutulan, T içinde bu olay sırasında kaybolacak yayı tutan yaprağa bir işaretçi ile beraber çemberin en alt noktasıdır.

Bütün nokta olayları sırasıyla bilinmektedir. Ancak çember olayları bilinmemektedir. Dolayısıyla çember olayının belirlenmesine ihtiyaç vardır.

Tarama çizgisi ilerledikçe her yeni olayda kıyı çizgisi topolojik yapısını değiştirecektir. Bu durum yeni sıralı yay üçlemelerinin oluşmasına ve daha önce var olan yay üçlemelerinin de kaybolmasına neden olabilir. Algoritmada her sıralı yay üçlemesinin potansiyel bir olay meydana getirebileceğinden olay kuyruğu Q'da tutulmalıdır. Burada dikkat edilmesi gereken iki ince nokta vardır. Birincisi sıralı üçlemenin iki kırılma noktasının yakınsamamasıdır. Bu hareket ettikleri doğrultulara bakılınca iki kırılma noktasının gelecekte buluşmayacağı anlamına gelir. Bu olayda kesişme noktasından uzaklaşan iki sektör boyunca hareket ettiklerinde oluşur. Bu durumda bu üçleme potansiyel bir çember olayı teşkil etmemektedir.

İkincisi ise bir üçlemenin yakınsayan kırılma noktaları olsa dahi buna karşılık gelecek çember olayının oluşması gerekmez. Üçleme kıyı çizgisinde yeni bir mevzinin görünmesiyle kaybolabilir. Bu durumda bu olaya yanlış alarm adı verilmektedir.

Sonuçta algoritmanın yaptığı özetlenirse, oluşan her sıralı yay üçlemesi kontrol edilir. Örneğin bir nokta olayında yeni üç adet üçleme oluşabilir. Birincisi eski üçlemenin solunda oluşan yay, biri ortadaki ve bir de üçlemenin sağında oluşan yay ile meydana gelen üçlemelerdir. Yeni bir üçleme yakınsayan kırılma noktalarına sahipse bu olay, Q olay kuyruğuna dâhil edilir. Bir nokta olayı sırasında oluşacak üç üçlemeden ortada olanının hiçbir zaman bir çember olayına neden olmayacağına dikkat edilmelidir. Çünkü sağ ve sol yaylar aynı parabolden geldikleri için kırılma noktaları uzaklaşacaktır. Bunun yanında kaybolan bütün üçlemelerin olay kuyruğunda bir girdisinin olup olmadığı da kontrol edilmelidir. Eğer varsa, bu bir yanlış alarm olacaktır ve olay kuyruğundan silinmelidir. Bu işlem T'deki yapraklardan olay kuyruğundaki karşılık gelen çember olayına işaret eden işaretçiler kullanılarak kolayca yapılabilir.

Yardımcı Teorem 3.1: Her Voronoi köşesi bir çember olayı sayesinde belirlenir.

Şimdi algoritma yapısı verilebilir. Bütün olaylar işlendiğinde ve olay kuyruğu boşaldığında henüz kıyı çizgisinin kaybolmadığına dikkat edilmelidir. Hala bulunan kırılma noktaları Voronoi diyagramının yarı doğrularına tekabül etmektedir. Daha

öncede değinildiği gibi yarı doğrular $DCEL$ yapısı tarafından tutulamayacağından bu kenarların bağlanacağı bütün sahneyi içine alan bir kutu eklenmelidir.

```
Algoritma FORTUNE_VORONOİ_DİAGRAMI (P)

Girdi: Düzlemde noktalar kümesi

Çıktı: D DCEL veri yapısında bir çevreleyen kutu içinde verilen P'nin
Voronoi diyagramı Vor(P)

   1. Bütün nokta olaylarını Q olay kuyruğuna dahil ederek Q'yu
      oluştur.

      Boş bir T durum yapısı oluştur.

      Boş bir D çift bağlı kenar listesi (DCEL) yapısı oluştur.

   2. while (Q boş olmadığı sürece)

   3. do (En büyük y koordinatına sahip olayı Q'dan çıkar)

   4. If ( Olay pi' de gerçekleşen bir nokta olayı ise)

   5. then NOKTA_OLAYI_GERÇEKLEŞTİR (pi)

   6. else ÇEMBER_OLAYI_GERÇEKLEŞTİR (γ) burada γ, kaybolacak yayı
      ifade eden T'nin yaprağıdır.

   7. T'nin içinde Voronoi diyagramının yarı doğrularına karşılık
      gelen iç düğüm noktaları bulunmaktadır. Voronoi diyagramının
      bütün köşelerini içerecek bir kutu hesapla ve yarı doğruları bu
      kutulara bağla. DCEL yapısını güncelleştir.

   8. DCEL'deki kenarları hücre kayıtlarını eklemek üzere ters çevir.
```

Olayları hesaplayan alt yordamlarda şu şekilde verilebilir.

```
NOKTA_OLAYI_GERÇEKLEŞTİR(pi)
```

1. Eğer T boş ise p_i'yi T'ye dahil edip dön. Aksi takdirde sonraki adımlarla devam et.

2. T'nin içinde p_i'nin üzerinde olan α yayını ara. Eğer α'yı ifade eden yaprak Q içinde bir çember olayına işaret eden bir işaretçiye sahipse bu çember olayı yanlış alarmdır ve bu olay Q'dan silinmelidir.

3. T'nin α'yı ifade eden yaprağını, üç yaprağa sahip bir alt dal ile değiştir. Ortadaki yaprak yeni nokta p_i'yi, diğer iki yaprak ise α yaprağında saklanan p_j noktalarını tutsun. Yani $\langle p_j, p_i \rangle$ ve $\langle p_i, p_j \rangle$ veri gruplarını yeni iki iç düğümde sakla. Gerekli ise iki kırılma noktasını ifade eden T'de dengeleme operasyonlarını gerçekleştir.

4. $V(p_i)$ ve $V(p_j)$ Voronoi hücrelerini bölecek yeni doğru kayıtlarını oluştur. (Yeni iki kırılma noktası tarafından çizilecek)

5. p_i için oluşan yeni yayın solda olduğu sıralı yay üçlemelerini kırılma noktalarını yakınsaması için kontrol et. Eğer varsa Q'ya bir çember olayı ve T ve Q'daki düğümlere karşılıklı olarak işaretçiler ekle. Aynı işlemi yeni yayın sağda olduğu yay üçlemesi için tekrarla.

ÇEMBER_OLAYI_GERÇEKLEŞTİR

1. T'den kaybolan α yayını ifade eden γ yaprağını sil. İç düğümlerde kırılma noktalarını tutan veri gruplarını güncelle. Gerekli ise T'de dengeleme operasyonlarını gerçekleştir. Q'dan α'yı içeren bütün çember olaylarını sil. Bu olaylar T'deki γ'ya bağlı ileri ve geri yönünü işaretçiler kullanılarak bulunabilir. (α'nın orta yay olduğu olay işlenmektedir ve Q'dan silinecektir.)

2. Olaya neden olan çemberin merkez noktasını bir Voronoi köşesi kaydı olarak Voronoi diyagramında saklayan DCEL'ye ekle. Kıyı çizgisinde yeni oluşan iki kırılma noktasına karşılık iki yarı doğru kaydı oluştur. Aralarındaki işretçileri oluştur. Bu köşe noktasında biten yarı doğru kayıtlarına üç yeni kayıt ekle.

3. Yeni sıralı yay üçlemelerinden eski sol yayın ortada olduğu üçlemenin kırılma noktalarının birbirine yakınsayıp yakınsamadıkları konusunda kontrol et. Eğer yakınsıyorlarsa Q'ya çember olayı ekle ve karşılıklı olarak T ve Q'daki işaretçileri ayarla. Aynı işlemi eski sağ yayın ortada olduğu üçleme için tekrarla.

Yardımcı Teorem 3.2 : Algoritma $O(n\log n)$ sürede çalışır ve $O(n)$ bellek kullanır.

İspat: T ağacındaki ve Q kuyruğundaki ekleme ve silme gibi temel işlemler $O(\log n)$ sürede yapılmaktadır. $DCEL$'deki temel işlemler ise sabit sürede yapılır. Bir olayın işlenmesi sırasında sabit sayıda temel işlem yapılmaktadır, dolayısıyla bir olayın işlenmesi $O(\log n)$ sürede gerçekleşir. Tam olarak n adet nokta olayı vardır. Çember olaylarının sayısının belirlenmesi için her Voronoi köşesinin bir çember olayı ile oluştuğu göz önüne alınabilir. Yanlış alarmlar işlenmeden silindiği için dikkate alınmasına gerek yoktur. Yanlış alarmlar gerçek bir olayın işlenmesi esnasında oluşturulup silindikleri için onlar için harcanan süre gerçek olayın işlenmesinde bastırılmış olur. Bu yüzden çember olaylarının sayısı en fazla $2n - 5$ olacaktır. Süre ve bellek ihtiyacı da bunlara bağlı olarak bulunur.

Son olarak bozulma olabilecek durumlara değinilecektir. Algoritma olayları yukarıdan aşağıya doğru işlemektedir, buna bağlı olarak iki veya daha fazla mevzinin aynı yatay doğru üzerinde bulunması durumunda bir bozulma gerçekleşebilir. Daha açık söylenirse, iki noktanın y-koordinatı aynı ise bu durum oluşabilir. Bu olaylar x-koordinatları ayrık ise herhangi bir sırada işlenebilir. Dolayısıyla aynı y-koordinatına ve farklı x-koordinatına sahip olaylar bir sorun oluşturmaz. Ancak bu durum algoritmanın başında gerçekleşirse yani ikinci noktanın y-koordinatı birinci noktanın y-koordinatıyla aynı ise özel bir koda ihtiyaç vardır. Çünkü ikinci noktanın üzerinde henüz bir yay yoktur.

Şimdi de çakışan olay noktaları olduğu düşünülsün. Örneğin aynı çember üzerinde dört veya daha fazla mevzinin olduğu durumda çakışan çember olayları olacaktır. Bu noktaların üzerinde bulunduğu çemberin içi boştur ve merkezi de bir Voronoi köşesi olacaktır. Bu köşenin derecesi en az dörttür. Bu gibi özel durumları belirlemek için ilave kod yazılabilir ancak buna gerek de yoktur. Çünkü algoritma bu olayları gelişigüzel sırada işler ve derecesi dört olan bir Voronoi köşesi oluşturmak yerine aynı koordinatlarda derecesi üç olan iki köşe noktası oluşturur. Bu köşeler arasındaki uzaklık sıfırdır. Bu bozuk kenar algoritmanın tamamlanmasından sonra silinebilir.

Olayların sırasının belirlenmesinde böyle bozulmalar olabileceği gibi bir olay işlenirken de bozulmalar olabilir. Bu durumda işlenen p_i mevzisinin tam olarak kıyı çizgisindeki iki yayın kırılma noktasının altında olması durumunda gerçekleşir. Bu yayların birinin uzunluğu sıfır olacaktır. Bu uzunluğu sıfir olan yay da bir çember olayı tanımlayan bir yay üçlemesinin orta yayı durumundadır ve bu çemberin de en alt noktası p_i ile çakışır. Algoritma bu çember olayını Q olay kuyruğuna ekler çünkü onu tanımlayan bir yay üçlemesi vardır. Bu olay işlendiğinde Voronoi diyagramının bir köşesi doğru olarak oluşturulur ve bu sıfır uzunluklu yay sonradan silinebilir. Diğer bir bozulma da kıyı çizgisi üzerindeki sıralı üç yayın aynı doğru üzerindeki üç mevzi tarafından tanımlanması durumunda gerçekleşir. Bu durumda bu mevziler bir çember tanımlamaz ve bir çember olayı oluşmaz.

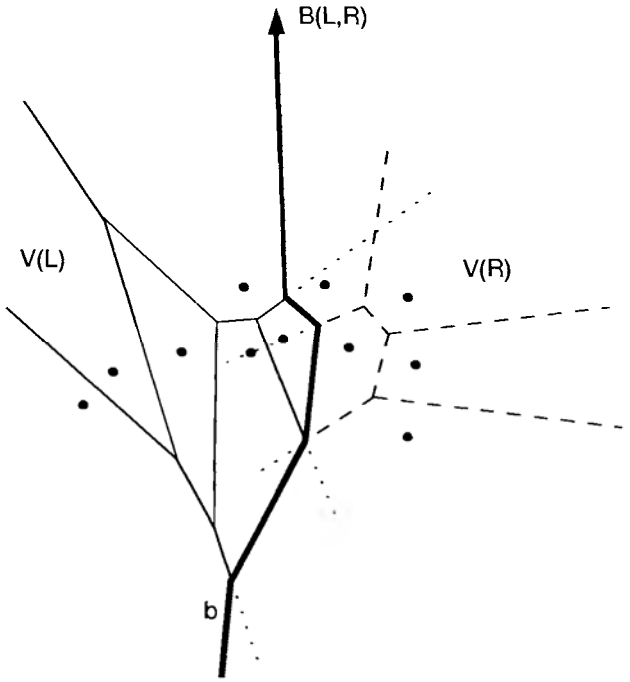
3.1.6 Böl ve Fethet (Divide & Conquer) algoritması

Voronoi diyagramının hesaplanması için ilk deterministik en kötü durum optimize edilmiş algoritma Shamos ve Hoey (1975) tarafından ortaya konulmuştur. Bu böl ve fethet algoritmasında P nokta mevzileri kümesi, aşağı yukarı aynı boyutta olan L ve R adında iki alt kümeye bir bölme çizgisi vastasıyla bölünür ve özyinelemeli olarak $Vor(L)$ ve $Vor(R)$ hesaplanır. Burada önemli olan nokta, bölme çizgisinin bulunması ve $Vor(L)$ ve $Vor(R)$'nin $Vor(P)$'yi oluşturmak üzere birleştirilmesidir. Bu işlem $O(n)$ sürede yapılabilirse toplam çalışma süresi $O(n\log n)$ olacaktır (Aurenhammer ve Klein 2000).

P kümesindeki mevziler önceden x ve y koordinatlarına göre sıralanırsa yatay ve dikey bölme çizgileri kolaylıkla bulunabilir.

Ayırma adımı, $Vor(P)$'nin bütün kenarları içinden L'nin Voronoi hücreleri ile R'nin Voronoi hücrelerini birbirinden ayıran $B(L,R)$'nin hesaplanmasından ibarettir.

Bölme çizgisinin dikey olduğu ve L'ninde bu çizginin solunda olan mevzilerin kümesi olduğu kabul edilsin.



Şekil 3.11 $\boldsymbol{Vor(L)}$, $\boldsymbol{Vor(R)}$ ve $\boldsymbol{B(L,R)}$'nin birleştirilerek $\boldsymbol{Vor(P)}$'nin oluşturulması

Yardımcı Teorem 3.3: $B(L,R)$'nin kenarları tek bir y-monoton poligonal zincir oluşturur. $Vor(P)$ içinde L'nin mevzilerinin hücreleri $B(L,R)$'nin solunda ve R'nin mevzilerinin hücreleri $B(L,R)$'nin sağına yer alır.

İspat: b, $B(L,R)$'nin herhangi bir kenarı ve $l \in L$ ve $r \in R$'de hücreleri b'ye komşu olan iki mevzi olsun. l, r'den daha küçük bir x koordinatına sahip olduğundan b yatay olamaz ve l'nin hücresi b'nin solunda olmalıdır.

Sonuçta $Vor(P)$; $Vor(L)$, $Vor(R)$ ve $B(L,R)$'nin birleştirilmesiyle elde edilebilir (Şekil 3.11). $B(L,R)$ sonsuzda bir başlangıç kenarı bulunarak ve $Vor(L)$ ve $Vor(R)$ boyuca izlenerek bulunabilir.

Teorem 3.1: Böl ve fethet algoritması düzlemde n nokta mevzisinden oluşan bir kümenin Voronoi diyagramını $O(n)$ bellek kapasitesi kullanarak $O(n\log n)$ sürede hesaplar.

3.2 Genelleştirilmiş Voronoi Diyagramları ve Uygulamaları

3.2.1 Doğru parçalarının Voronoi diyagramları

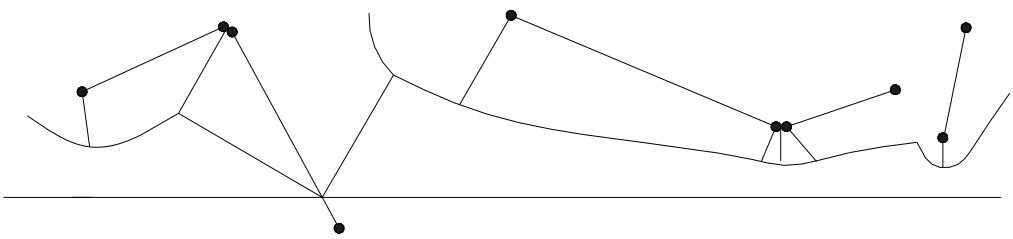
Voronoi diyagramı noktaların yanında nesneler içinde tanımlanabilir. Düzlemdeki bir noktanın bir nesneye olan uzaklığı, nesnenin üzerinde o noktaya en yakın nokta kullanılarak belirlenir. Basit olarak iki nokta arasındaki açıortay bir doğru tanımlamaktadır, iki ayrık doğru parçasının açıortayı ise daha karmaşık bir şekle sahip olacaktır. Bu şekil yediye kadar parça ihtiva eder. Her parça, doğru parçası ya da parabolik yay olabilir. Parabolik yaylar doğru parçasının birindeki en yakın nokta uç noktası ve diğerindeki en yakın nokta bir iç nokta olduğunda oluşur. Diğer bütün durumlarda oluşacak parçalar düzdür. Açıortaylar ve dolayısıyla Voronoi diyagramı daha karmaşık bir yapıya sahip olmasına rağmen n adet ayrık doğru parçasının Voronoi diyagramının kenarlarının, köşelerinin ve yüzlerinin sayısı sadece $O(n)$'dir (de Berg vd. 2008).

Doğru parçalarının kesişmediği ancak ortak uç noktalarına sahip oldukları düşünülsün. Bu durumda düzlemin belli bir bölgesinin tamamı ortak bir noktaları oldukları için her iki doğru parçasına da eşit uzaklıkta olacaktır ve açıortaylarda artık eğri bile olmayacaktır. Ortak uç noktaları olan doğru parçalarının Voronoi diyagramlarının tanımlanması ve hesaplanmasındaki bu karmaşıklıktan kaçınmak için bütün doğru parçalarının tamamen ayrık oldukları kabul edilecektir. Birçok uygulamada ayrık doğru parçaları elde etmek üzere doğru parçaları çok küçük bir miktar kısaltılabilir.

Tarama çizgisi algoritması doğru parçalarının Voronoi diyagramının hesaplanmasına adapte edilebilir. $S = \{s_1, s_2, \ldots\ldots\ldots, s_n\}$ ayrık n doğru parçasından oluşan bir küme olsun. Burada S'deki doğru parçaları yine mevzi olarak adlandırılacaktır.

Nokta mevzileri için geliştirilmiş tarama çizgisi algoritması düşünüldüğünde kıyı çizgisi, tarama çizgisinin üzerindeki en yakın mevzi ile tarama çizgisine eşit uzaklıktaki noktaların oluşturduğu parçalı parabolik bir eğri olarak söylenebilir. Peki, mevziler

doğru parçaları olduklarında kıyı çizgisi nasıl olacaktır? İlk göze çarpan bir doğru parçası kısmen tarama çizgisinin üstünde ve kısmen altında olabilir. Bu durumlarda bu mevzilerin sadece tarama çizgisi üstünde kalan kısmı dikkate alınacaktır. Bu yüzden l tarama çizgisinin herhangi bir pozisyonunda, kıyı çizgisi, bir mevzinin tarama çizgisi üzerinde kalan kısmının en yakın noktası ile tarama çizgisine eşit uzaklıktaki noktalardan oluşacaktır. Bu kıyı çizgisinin doğru parçalarından ve parabolik yaylardan oluşacağı anlamına gelmektedir. Parabolik bir yay sadece mevzinin en yakın noktasının uç noktası olması durumunda gerçekleşir. Eğer mevzinin kıyı çizgisine en yakın noktası içteki bir nokta ise kıyı çizgisinin o bölümü bir doğru parçasından oluşur. Eğer bir mevzinin iç bir noktası l'yi keserse kıyı çizgisinde bir uçları bu kesişim noktasında birleşen iki doğru parçası oluşur (Şekil 3.12).



Şekil 3.12 Doğru parçalarının Voronoi diyagramının oluşturulması

Kıyı çizgisini oluşturan parabolik yayların ve doğru parçalarının arasındaki kırılma noktaları çok çeşitli yollarla oluşabilir.

1. Eğer bir p noktası, iki mevzi ve l doğrusuna eşit uzaklıkta iken bu iki mevzinin uç noktalarına en yakın nokta ise p bir doğru parçası oluşturacak bir kırılma noktasıdır.
2. Eğer bir p noktası, iki mevzi ve l doğrusuna eşit uzaklıkta iken bu iki mevzinin iç noktalarına en yakın nokta ise p bir doğru parçası oluşturacak bir kırılma noktasıdır.

3. Eğer bir p noktası, iki mevzi ve l doğrusuna eşit uzaklıkta iken bu iki mevzinin birinin uç noktasına ve birinin iç noktasına en yakın nokta ise p bir parabolik eğri oluşturacak bir kırılma noktasıdır.
4. Eğer p bir mevzinin uç noktasına en yakın nokta ise ve bu nokta ile arasında en yakın mesafe dikey bir çizgi ise p bir doğru parçası oluşturan bir kırılma noktasıdır.
5. Eğer bir mevzi tarama çizgisini keserse, kesişim noktası bir doğru parçası oluşturacak bir kırılma noktasıdır.

Dördüncü ve beşinci tipteki kırılma noktaları aslında Voronoi diyagramının bir yayını izlemez. Çünkü sadece bir mevziyi kapsamaktadır. Ancak algoritmanın doğru çalışması için bu tip kırılma noktaları ve karşılık gelen olayların da dikkate alınması gerekmektedir.

Nokta mevzileri için geliştirilen tarama çizgisi algoritmasında olduğu gibi bu algoritmada da nokta olayları ve çember olayları vardır. Bir nokta olayı tarama çizgisi bir mevzinin uç noktalarına ulaştığında gerçekleşir. Mevzilerin üst uç noktalarında gerçekleşen nokta olayları, alt uç noktalarında gerçekleşen nokta olaylarından farklı incelenmelidir. Üst uç noktasında kıyı çizgisinin yayı ikiye bölünür ve arada dört yay oluşur. Bu dört yay arasında oluşan kırılma noktaları yukarıda belirtilen son iki tiptedir. Alt uç noktasında ise mevzinin iç noktasıyla tarama çizgisinin kesişimindeki kırılma noktası aralarında parabolik bir yayın olduğu dördüncü tipteki iki kırılma noktası ile değiştirilir.

Benzer şekilde çeşitli tipte çember olayları mevcuttur. Bunların hepsi kıyı çizgisinin bir yayının kaybolması durumuna karşılık gelir ve tarama çizgisi iki ya da üç mevzi ile tanımlanan boş çemberin en alt noktasına ulaştığında gerçekleşir. Bu boş çemberlerin merkez noktası iki ardışıl kırılma noktasının buluşacağı noktalardır. Buluşacak kırılma noktalarının türüne bağlı olarak farklı durumlar belirlenip incelenebilir. Eğer iki kırılma noktası ilk üç türden herhangi birinde ise o zaman üç mevzi içerilmektedir. Eğer kırılma noktalarından biri dördüncü tipte ise o zaman sadece iki mevzi içerilir. Beşinci tipteki kırılma noktalarında ayrık doğru parçaları içerilmez.

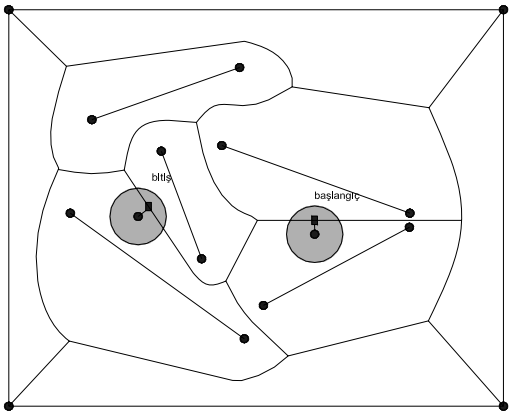
Algoritmanın hesapladığı Voronoi diyagramı düz kenarlar ve parabolik yaylarda oluşan bir alt bölümlemedir. Böyle bir bölümleme $DCEL$ veri yapısında tutulabilir mi? Bu mümkündür, hatta yapı hiç yeni düzenlemeye ihtiyaç duymadan kullanılabilir. Her yüz için karşılık gelen mevzi saklanır ve herhangi bir e yarı doğrusu için e'nin açıortayı olduğu iki mevzi bulunabilir. Kenarın arasında yer aldığı iki köşe noktası da kolaylıkla belirlenebildiğinden herhangi bir kenarın şekli sabit sürede bulunabilir.

Bu tarama çizgisi algoritması, nokta mevzileri için olan versiyonunun bir uzantısıdır. Sadece daha fazla durum belirlenmeli ve işlenmelidir. Ancak algoritma yine $O(n)$ olaya sahiptir ve her olayda $O(\log n)$ sürede işlenebilir.

Teorem 3.2: n noktadan oluşan bir ayrık doğru parçalarının kümesinin Voronoi diyagramı $O(n\log n)$ sürede $O(n)$ bellek kapasitesi kullanılarak hesaplanabilir.

Doğru parçalarının Voronoi diyagramlarının önemli bir uygulaması hareket planlamadır. Bir R robotu ve engellerin oluşturduğu n adet doğru parçası olduğu düşünülsün. Robotun her yöne rahatlıkla hareket edebildiği ve bir D diski ile ifade edilebildiği kabul edilsin. Robotun bir noktadan diğer noktaya engellere çarpmadan gidebileceği yolu veya böyle bir yolun olmadığı belirlenmeye çalışılsın.

Bir hareket planlama tekniği retraksiyon olarak adlandırılır. Retraksiyonda temel prensip doğru parçaları arasında tam ortadaki Voronoi diyagramı yaylarını izlemektir (Şekil 3.13). Böylelikle en açık yol bulunmuş olur. Dolayısıyla Voronoi diyagramının eğrileri üzerinde oluşturulacak bir yol çarpışma olmadan ilerlemek için en iyi seçenektir.

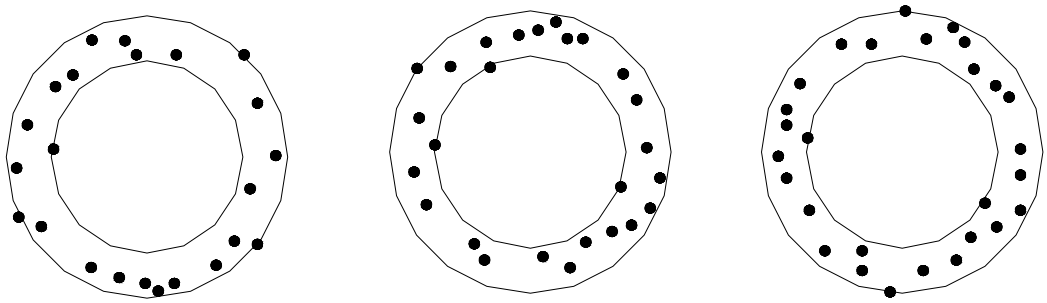


Şekil 3.13 Retraksiyon yöntemi

Doğru parçalarının Voronoi diyagramlarının diğer uygulama alanları arasında biyoloji, coğrafya, doku tanıma, bilgisayar grafikleri sayılabilir. (Kirkpatrick 1979, Lee 1982)

3.2.2 En uzak nokta Voronoi diyagramları

En uzak nokta Voronoi diyagramları Voronoi diyagramlarının kullanıldığı diğer bir uygulama alanıdır. Bu uygulama alanı için bir örnek verilmesi gerekirse, nesnelerin mükemmel derecede yuvarlak olması istendiği durumlarda bu nesneler yuvarlaklık testine tabi tutulur. Bu nesnelerden alınan örnek noktalar yaklaşık olarak bir çember üzerinde yer alan bir P noktalar kümesi elde edilir. Bu noktalar kümesinin yuvarlaklığı bu noktaları ihtiva eden en küçük genişliğe sahip halka ile ifade edilir. Bu halka eşmerkezli iki çemberin arasında kalan bölgedir ve halkanın genişliği bu iki çemberin yarıçapları arasındaki farktan ibarettir (de Berg vd. 2008).



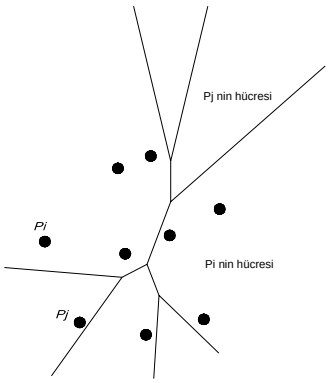
Şekil 3.14 Bir nokta kümesi için en küçük genişlikteki halka durumları

En küçük genişlikteki halka sınır çemberleri üzerinde *P* noktalar kümesinin bazı noktalarını barındırmalıdır. Durum analiz edildiğinde, dış çemberin üzerinde en az bir nokta olması gerektiği açıktır, aksi takdirde çemberin çapı azaltılabilir. Benzer şekilde iç çemberin de üzerine en az bir nokta olmalıdır, yoksa çemberin çapı arttırılabilir. Ancak her iki sınır çember üzerindeki birer nokta olması en küçük genişlikteki halkayı tanımlamamaktadır. Bu halkayı tanımlayabilecek üç durum mevcuttur ve bu durumlarda iki çember üzerinde toplam dört nokta bulunmaktadır (Şekil 3.14) :

- Dış çember *P*'nin en az üç noktasını ve iç çember *P*'nin en az bir noktasını içerir.
- Dış çember *P*'nin en az bir noktasını ve iç çember *P*'nin en az üç noktasını içerir.
- Dış çember ve iç çemberin her ikisi *P*'nin en az iki noktasını içerir.

Eğer dış çember ve iç çember listelenen durumlardan birinden daha az nokta içeriyorsa her zaman daha küçük genişliğe sahip bir halka bulunabilir. Bu en küçük genişlikteki halkanın bulunması problemi bu halkanın merkezinin bulunması ile çözülebilir. Merkez nokta bulunduktan sonra halka *P*'nin bu merkez noktaya en uzak ve en yakın noktalarının belirlenmesi ile tanımlanabilir. Eğer *P* noktalar kümesinin Voronoi diyagramı hesaplanırsa, en yakın nokta merkez noktanın içinde olduğu Voronoi hücresi ile bulunabilir. Benzer bir yapı en uzak nokta için de oluşturulabilir ve bu yapı en uzak nokta Voronoi diyagramı olarak adlandırılmaktadır. Bu yapıda düzlem *P*'nin aynı

noktasının en uzak nokta olduğu hücrelere bölünür. Bir p_i noktasının en uzak nokta Voronoi hücresi, standart Voronoi hücresinde olduğu gibi $n-1$ yarı-düzlemin kesişimidir. Ancak bu sefer dik açıortayların diğer tarafı yani p_i'den uzakta olan tarafı alınır. Bu yüzden en uzak nokta Voronoi diyagramlarının bütün hücreleri konvekstir. P noktalar kümesindeki her noktanın en uzak nokta Voronoi diyagramı içinde bir hücresi bulunmaz; yarı-düzlemlerin kesişimi boş olabilir (Şekil 3.15). Düzlemdeki herhangi bir noktaya P kümesi içindeki en uzak noktanın konveks zarf üzerinde bulunması gerektiğini görmek zor değildir. Bu yüzden konveks zarf içine kalan bir noktanın en uzak nokta Voronoi diyagramında bir hücresi yoktur.



Şekil 3.15 En uzak nokta Voronoi diyagramı örneği

Yardımcı Teorem 3.4: Düzlemde verilen bir P noktalar kümesi için; P'nin herhangi bir noktasının en uzak nokta Voronoi diyagramında bir hücresi olabilmesi için gerek ve yeter şart bu noktanın P'nin konveks zarfının bir köşe noktası olmasıdır.

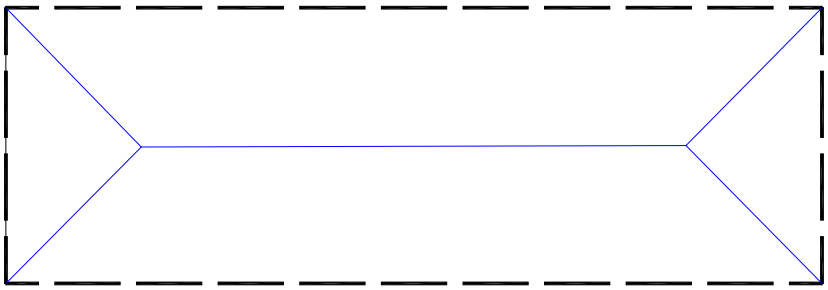
En uzak nokta Voronoi diyagramı artırımlı bir algoritma ile hesaplanabilir. Önce P noktalar kümesinin konveks zarfı hesaplanır, köşeleri belirlenir ve bu köşeler rastgele sırada bir listede tutulur. Sonra bu listeden ilk üç nokta hariç diğer noktalar çıkarılır ve kalan üç nokta için en uzak nokta Voronoi diyagramı hesaplanır. Sonra da kalan noktalar sırayla eklenerek en uzak nokta Voronoi diyagramı oluşturulmaya devam edilir.

Teorem 3.3: Düzlemde verilen n noktadan oluşan bir noktalar kümesinin en uzak nokta Voronoi diyagramı $O(nlogn)$ sürede $O(n)$ bellek kapasitesi kullanılarak hesaplanabilir (de Berg vd. 2008).

3.2.3 İç eksen

Bu kısımda Voronoi diyagramının en basit genelleştirmesi olan iç eksen incelenecektir. Bu genelleştirme noktalar kümesinin sonsuz bir noktalar kümesi olmasına izin vermektedir, bu noktalar kümesi bir poligonun sınırını oluşturur (O'Rourke 1997).

Voronoi diyagramının tanımından yola çıkılırsa, bu diyagramı oluşturan noktalar kümesindeki noktaların en yakın mevzisi tek değildir. Bu noktalar iki veya daha fazla mevziye eşit uzaklıktadır. Bir poligonun iç ekseni (poligonun simetrik ekseni veya iskeleti olarak da adlandırılabilir) poligonun sınırındaki noktaların birden fazlasına eşit uzaklıktaki noktalar kümesi olarak tanımlanabilir.



Şekil 3.16 Bir dikdörtgenin iç ekseni

Bir dikdörtgenin iç ekseni Şekil 3.16'da gösterilmiştir. Orta eksenin yatay parçası üzerindeki her nokta dikdörtgenin üst ve alt kenarlarındaki noktalardan eşit uzaklıktaki noktalardır. Benzer şekilde çapraz parçalar ardışıl kenarlardan ve parçaların kesişim noktaları da üç kenardan eşit uzaklıktaki noktalardan oluşmaktadır.

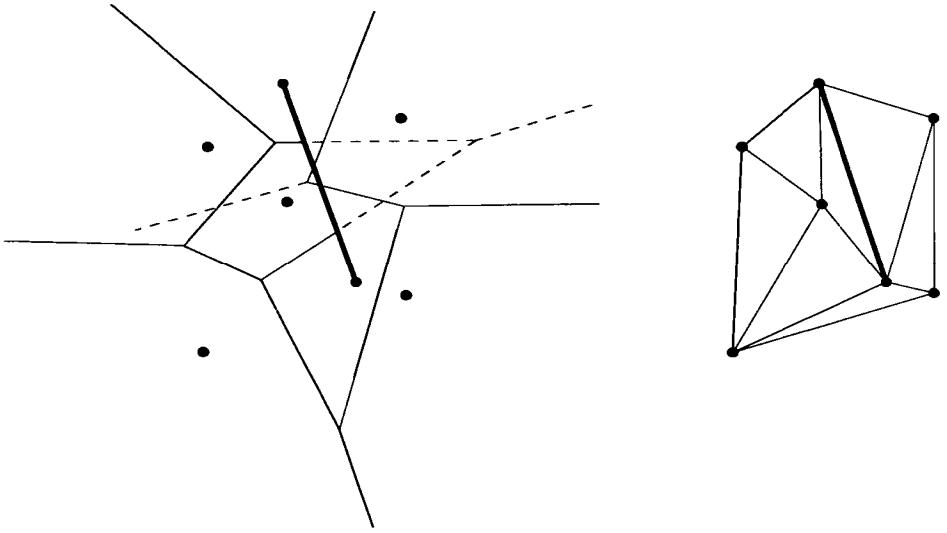
3.2.4 Kısıtlanmış Voronoi diyagramı ve Delaunay üçgenlemesi

Belirli durumlarda, bir mevzi kümesi üzerinde herhangi bir kısıtlama olmadan elde edilen Voronoi diyagramı ve Delaunay üçgenlemesi pratik ihtiyaçları karşılamak için yeterli olmamaktadır. Birbirlerine geometrik olarak yakın olan iki mevzinin komşu sayılmaması için sebeplerin olduğu durumlar vardır. Örneğin birbirine coğrafik olarak yakın iki şehir bir dağ sırasıyla birbirinden ayrılmışsa, bir kamyon sürücüsünün bakış açısından çok uzak görünür. Ya da bu iki şehir yakın olmalarına rağmen farklı ülkelerin şehirleri olabilir.

P düzlemde verilmiş n adet noktadan oluşan bir küme ve L'de P tarafından kapsanan ve kesişmeyen doğru parçaları kümesi olsun. L kümesindeki doğru parçaları Voronoi diyagramının oluşturulması sırasında karşılaşılan engeller olacaktır. p_i ve p_j olarak isimlendirilen iki nokta arasındaki kısıtlanmış uzaklık aşağıdaki gibi tanımlanabilir (Aurenhammer and Klein 2000).

$$b(p_i, p_j) = \begin{cases} dist(p_i, p_j), & \text{eğer } \overline{p_i\, p_j} \cap L \neq \emptyset \text{ ise,} \\ \infty, & \text{diğer durumlarda.} \end{cases} \tag{3.2}$$

Sonuçta elde edilen kısıtlanmış Voronoi diyagramı $Vor(P, L)$'de birbirine yakın olan ancak birbirlerinden görülemeyen mevzi bölgeleri L içindeki doğru parçaları tarafından kırpılmıştır. Doğru parçalarının uç noktalarının Voronoi hücreleri, ilgili doğru parçası yakınında konveks değildir. Bu durum Şekil 3.17'de görülmektedir.



Şekil 3.17 Bir doğru parçası ile kısıtlanmış bir nokta kümesinin Voronoi diyagramı ve Delaunay üçgenlemesi

$Vor(P,L)$'nin çifteşi L kümesi içindeki doğru parçaları içerilse dahi P'nin tam bir üçgenlemesi değildir. Ancak $Vor(P,L)$ bir üçgenleme elde etmek için çifteş dönüşümü yapmak üzere modifiye edilebilir. Bu kısıtlama altında elde edilen üçgenleme olabildiğince Delaunay üçgenlemesi olacaktır.

Burada yapılacak modifikasyon oldukça basittir. Her $l \in L$ için l tarafından sağdan kırpılan bölgeler, sadece bu bölgelerin mevzileri varmış gibi sola doğru genişletilir. Benzer şekilde l tarafından soldan kesilen bölgeler de sağa doğru genişletilir. Bu durumda tabii ki hücreler üst üste gelebilir ve düzlemin belli bir kısmını tanımlamaz. Şimdi bir kenarı paylaşan hücrelerin mevzileri çifteş dönüşümüne tabi tutulursa, L'nin içerildiği tam bir üçgenleme elde edilmiş olur. Bu üçgenlemeye de kısıtlanmış Delaunay üçgenlemesi denir ve $Del(P,L)$ şeklinde gösterilir. Kısıtlanmış Delaunay üçgenlemesinin kenar sayısı en fazla $3n-6$ olacaktır. Hem kısıtlanmış Voronoi diyagramlarının hem de kısıtlanmış Delaunay üçgenlemelerinin boyutları doğrusaldır.

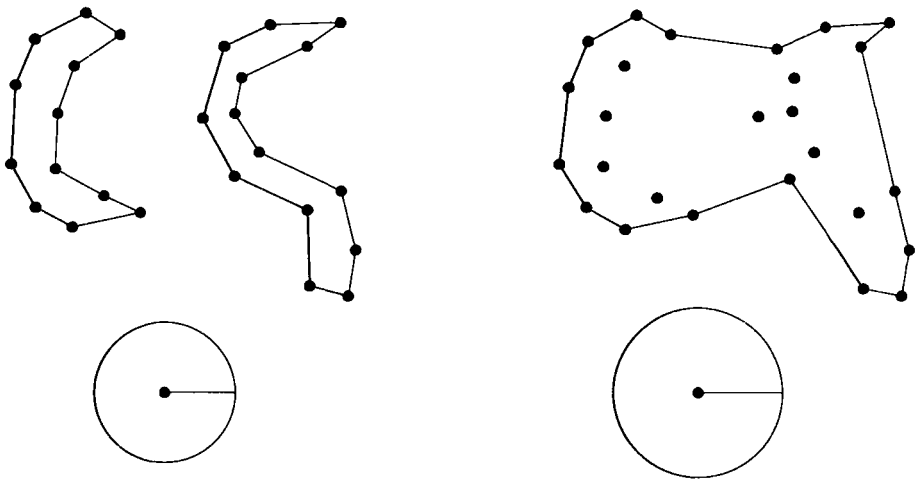
$Vor(P,L)$ ve $Del(P,L)$'nin hesaplanması için birçok yayında değişik algoritmalar sunulmuştur (Seidel 1988, Kao ve Mount 1992). Bu tez çalışması kapsamında geliştirilen yazılımda kısıtlanmış Voronoi diyagramı ve kısıtlanmış Delaunay

üçgenlemesi hesaplaması yapan algoritmalar tarama çizgisi algoritmasında değişiklikler yapılarak elde edilmiştir.

3.2.5 α Şekilleri

Düzlemdeki bir P nokta mevziler kümesinin şeklinin çıkarılması işlemi günümüzde, örneğin veri görselleştirilmesi ve doku tanıma gibi alanlarda önemli bir problem olarak ortaya çıkmaktadır. Belli bir mertebeye kadar P'nin şeklini P'nin konveks zarfı yansıtmaktadır. Konveks zarfın kenarları Delaunay üçgenlemesi $Del(P)$'nin bir kısmıdır. α-şekli kavramı ilk olarak Edelsbrunner tarafından ortaya konulmuştur (Edelsbrunner vd. 1983, Edelsbrunner 1987). α-şekilleri P'nin konveks zarfının genelleştirilmesi ile elde edilmektedir ve yine Delaunay üçgenlemesinin bir alt grafiği olma özelliğini korumaktadır.

$\alpha > 0$ için P'nin α-şekli $\alpha(P)$ şu şekilde tanımlanabilir. $p, q \in P$ mevzileri için α yarıçapına sahip ve sınırında p ve q noktalarını barındıran boş bir çember varsa p ve q noktalarını birleştiren kenar $\alpha(P)$ içinde yer alır. $\alpha(P)$ her zaman $Del(P)$'nin bir kısmıdır ve bu yüzden n ile orantılı sayıda kenara sahiptir (Aurenhammer and Klein 2000).



Şekil 3.18 Aynı nokta kümesinin iki farklı α değeri için α-şekli

α'nın çok büyük değerleri için P'nin konveks zarfı elde edilir. α değeri küçüldükçe P'nin şekli için daha iyi bir çözünürlük sağlanır. Eğer α'nın değeri P içindeki en yakın mevzi çifti arasındaki uzaklıktan daha küçükse $\alpha(P)$ hiçbir kenar içermez. Şekil 3.18'de bir noktalar kümesinin iki farklı α değeri için elde edilen α-şekli gösterilmektedir.

$\alpha(P)$ tanımı α'nın negatif değerlerine genişletilebilir. $-\alpha$ yarıçapına sahip diskler P'nin tamamını içerir. Bu durumda elde edilen şekil, P'nin konveks zarfından daha kaba bir şekildir ve $\alpha(P)$'nin kenarları, P'nin en uzak nokta Voronoi diyagramının çifteşi ile elde edilen üçgenleme üzerindedir.

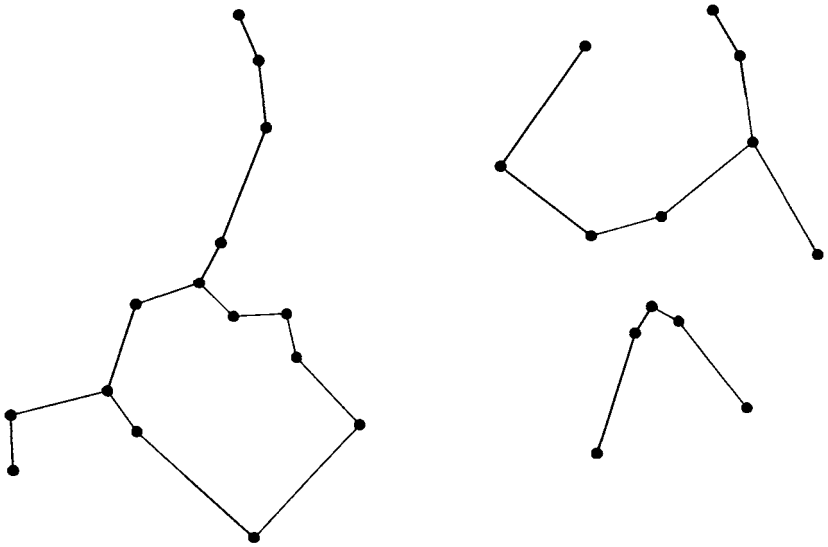
$-\infty < \alpha < \infty$ için P'nin bütün α-şekillerinin bütün kenarları $O(n)$ boyutunda iki üçgenleme tarafından içerilir. Her üçgenleme kenarı e için, e'yi içeren $\alpha(P)$'leri oluşturan α değerlerini kapsayan bir aktivite aralığı tanımlanabilir. Bu aktivite aralıkları $O(nlogn)$sürede hesaplanabilir ve $O(n)$ sürede verilen bir α değeri için $\alpha(P)$'nin elde edilmesi imkanı oluşur.

α-şekillerinin diğer bir önemli özelliği esnek bir yapıya sahip olmasıdır. Delaunay üçgenlemesi ile ilişkisi sayesinde α-şekli kavramı daha yüksek boyutlara genelleştirilebilir. Üç boyutlu uzaydaki α-şekilleri özel ilgi çekmektedir ve birçok bilim ve mühendislik alanında kullanılmaktadır (Edelsbrunner ve Mücke 1994).

3.2.6 β İskeletleri

Boş çemberlerle tanımlanan ve $Del(P)$'nin bir alt grafiği olan diğer bir yapı da P nokta mevzisi kümesinin β iskeletidir. Minimum ağırlıklı üçgenlemeler ile ilişkisinden ötürü giderek daha fazla ilgi çeken bu yapının hem çember bazlı hem de yarımay bazlı uyarlamaları bulunmaktadır (Kirkpatrick ve Radke 1985).

α-şekillerinin tersine β-iskeletini tanımlayan çemberin yarıçapı sabit değildir ve P noktalar kümesindeki mevziler arasındaki uzaklıklara bağlıdır. Şekil 3.19'da bir noktalar kümesinin $\beta = 1.2$ değeri için β-iskeleti gösterilmektedir.



Şekil 3.19 β = 1.2 için β-iskeleti

$p, q \in P$ olsun. $\beta \geq 1$ için eğer yarıçapı $\beta . dist(p, q)$ olan çemberin üzerinde p ve q noktaları varsa ve P'nin başka bir mevzisini içermiyorsa o zaman $\overline{pq}$ kenarı P'nin β-iskeleti içinde yer alır (Aurenhammer ve Klein 2000).

Eğer $\beta > \beta'$ ise $\beta(P)$, $\beta'(P)$'nin bir alt grafiğidir. $\beta = 1$ için P'nin Gabriel grafiği elde edilir. $\beta(P)$ bağlantısız bir grafik olabilir ancak Gabriel grafiği daima bağlantılıdır ve P'nin en küçük kapsar ağacını bir alt grafik olarak içermektedir. Gabriel grafiği $Del(P)$'nin sadece kendi çifteşleri olan Voronoi kenarlarını kesen kenarlarından oluşmaktadır. Bu, Gabriel grafiğinin ve β-iskeletinin oluşturulabilmesi için $O(n\log n)$ süreli bir algoritma ile hesaplanabileceği anlamına gelmektedir (Jaromczyk vd. 1992, Lingas 1994). Gabriel grafiğinin coğrafik verilerin işlenmesinde kullanışlı bir araç olduğu çeşitli kaynaklarda ortaya konulmuştur (Gabriel ve Sokal 1969, Matula ve Sokal 1980).

3.2.7 3-Boyutlu uzayda Voronoi diyagramı ve Delaunay üçgenlemesi

Voronoi diyagramlarının ve Delaunay üçgenlemelerinin en açık genelleştirmelerinden biri de $d \geq 3$ için d-boyutlu uzay genelleştirmesidir. Bu genelleştirmede Voronoi

diyagramının bazı temel özellikleri korunurken (örneğin hücrelerin konveks olması gibi), bazı temel özellikler de kaybolur (Örneğin lineer boyut gibi). d-boyutlu uzaydaki Voronoi diyagramları $(d+1)$ boyutlu uzaydaki geometrik nesnelerle yakından ilişkilidir (Aurenhammer ve Klein 2000).

Öncelikle 3-boyutlu uzayda Voronoi diyagramları üzerinde durulacaktır. $p, q \in P$ olan iki mevzinin dik açıortayı $\overline{pq}$ doğru parçasının orta noktasından geçen dik düzlem olacaktır. Bu durumda $p \in P$'nin Voronoi hücresi $V(p)$ bu dik açıortaylarla sınırlandırılmış yarı uzayların kesişiminden oluşmaktadır ve bu yüzden 3 boyutlu konveks bir çokyüzlüdür. $V(p)$'nin sınırları yüzlerden, kenarlardan ve köşelerden oluşmaktadır. Bu hücreler, yüzler, kenarlar ve köşeler 3-boyutlu uzayda bir hücre kompleksi meydana getirir. Bu hücre kompleksi yüz-yüze bir yapıdır. Eğer iki hücre boş olmayan bir f kesişimine sahipse f her iki hücrenin parçası olan bir yüz, kenar veya köşedir.

$V(p)$'nin yüzlerinin sayısı en fazla $n-1$'dir ve bunların da her biri bir $q \in P \setminus \{p\}$ içindir. Euler'in formülüne göre $V(p)$'nin kenar ve köşe sayısı da $O(n)$'dir. Bu durumda $Vor(P)$'nin 3-boyutlu uzayda toplam eleman sayısı $O(n^2)$ olacaktır. Ancak birim kürenin içinde düzenli bir yapıda alınan rastgele noktaların beklenen boyutunun $O(n)$ olduğu ortaya konulmuştur (Dywer 1991). Bu sonuç birçok pratik durumda yüksek boyutlu Voronoi diyagramlarının boyutlarının $O(n^2)$'den küçük olabileceğini göstermektedir.

2-boyutlu uzayda olduğu gibi 3-boyutlu uzaydaki Delaunay üçgenlemesi $Del(P)$ Voronoi diyagramının grafiksel çifteşidir. $Del(P)$, $Vor(P)$'nin her yüzü için bir kenar, her kenar için bir üçgen ve her köşe için bir dörtyüzlü içerir. Benzer şekilde $Del(P)$ en büyük boş çember özelliği ile de tanımlanabilir.

3-boyutlu $Vor(P)$'nin oluşturulması için değişik yöntemler ortaya konulmasına rağmen bunlardan en güvenilir ve geliştirilmesi kolay olanı rastgele artırımlı algoritmalardır. Bu

metotta yeni eklenen p mevzisinin hücresinin bütün yüzleri belirlenir ve bu hücre içinde kalan $Vor(P)$ elemanları silinir.

Çifteş düzleminde $Del(P)$'nin hesaplanmasında ise bu sefer eklenen p mevzisinin en büyük boş çemberi içinde kalacak bütün dörtyüzlülerin belirlenmesi ve silinmesi işlemi gerçekleştirilir. Daha sonra bu boş delik noktaya p mevzisi yerleştirilir.

4. MATERYAL ve YÖNTEM

4.1 Genel Bilgi

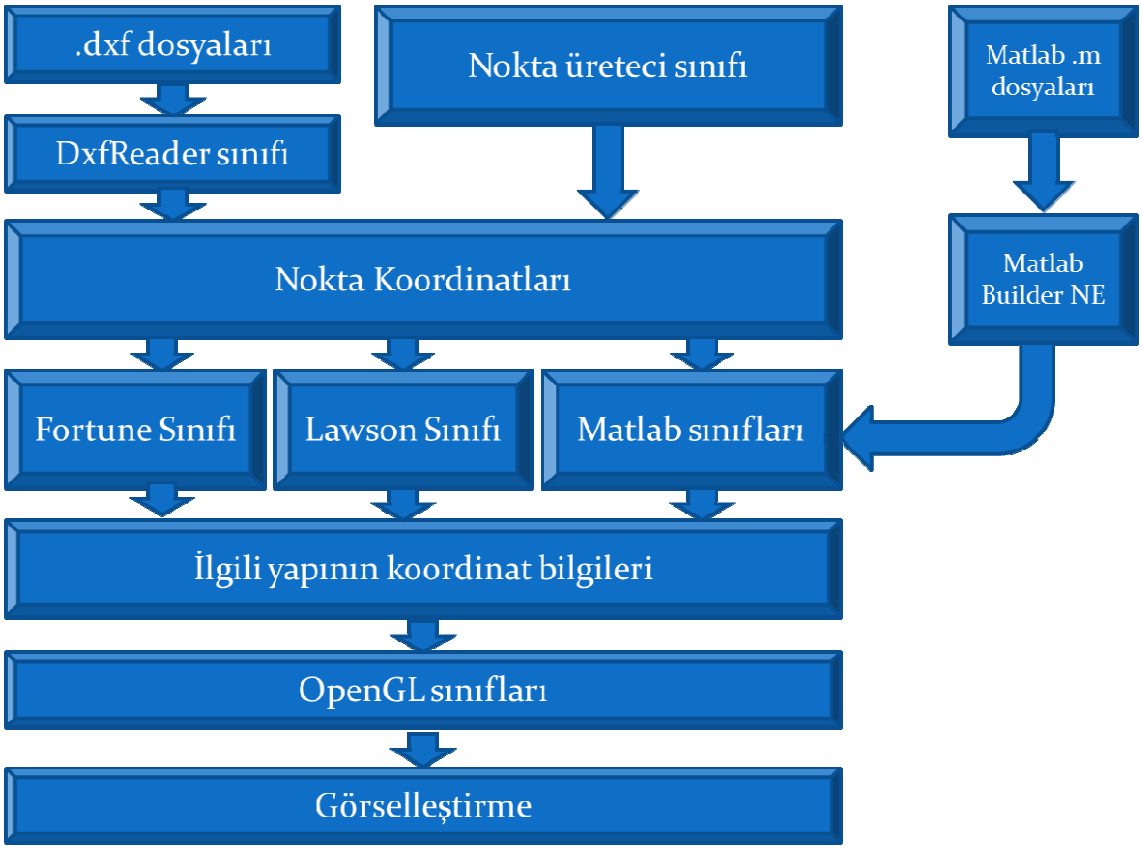
Tez kapsamında geliştirilen VVD3 yazılmı 2 ve 3 boyutlu Voronoi diyagramlarının oluşturulması ve Delaunay üçgenlemelerinin gerçekleştirilmesi için tasarlanmış ve tez içerisinde incelenen birçok genelleştirilmiş Voronoi diyagramı ve genelleştirilmiş Delaunay üçgenlemesi konusunun görselleştirilmesi sağlanmıştır. Program .NET ortamında C# programlama diliyle geliştirilmiş olmakla beraber görüntüleme işlemi için OpenGL kütüphanelerinden ve çeşitli algoritmaların kullanılması için de Matlab programının .NET ortamı için fonksiyonları paketleyen Builder NE aracından faydalanılmıştır. Ayrıca, C++ dilinde yazılmış CGAL kütüphanelerinin bazı bölümleri, algoritmaların geliştirilmesinde kullanılmıştır (http://cgal.org 2009).

Çizimlerin görüntülenmesi için OpenGL kütüphanelerinin tercih edilmesinin sebebi çeşitli çizimlerin Standart GDI kütüphaneleriyle gerçekleştirilmesinde karşılaşılan zorluklardır. OpenGL ile ilgili temel düzeyde bilgi EK 1'de verilmiştir.

Programda bazı Matlab fonksiyonları da kullanılmıştır. Böylelikle daha fazla içerik programa dahil edilebilmiştir. Matlab fonksiyonlarının .NET ortamında kullanılmasını sağlayan Builder NE kullanımı ile ilgili bilgiler EK 2'de verilmiştir.

VVD3 programı 2-boyutlu ve 3-boyutlu işlemler olmak üzere temel olarak iki kısımdan oluşmaktadır. Her iki kısımda da tez içerisinde incelenen birçok konunun uygulaması gerçekleştirilmiştir.

VVD3 programını oluşturan sınıfların ilişkileri Şekil 4.1'de gösterilmektedir. Voronoi diyagramı ya da Delaunay üçgenlemesi yapılacak nokta kümesinin veya poligonların programa aktarılması için nokta üreteci sınıfı kullanılabileceği gibi dxf dosya formatını okuyabilen sınıf da kullanılabilir. Bu bilgi daha sonra ilgili algoritmaya gönderilerek sonuç koordinat bilgileri elde edilmektedir. Sonra da bu koordinat bilgileri OpenGL'e gönderilerek görselleştirme işlemi yapılmaktadır.

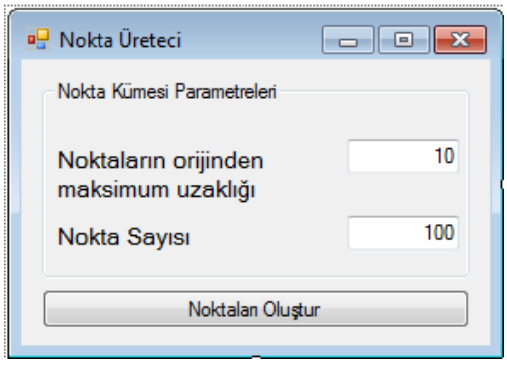


Şekil 4.1 VVD3 yazılımı sınıf hiyerarşisi

5. BULGULAR

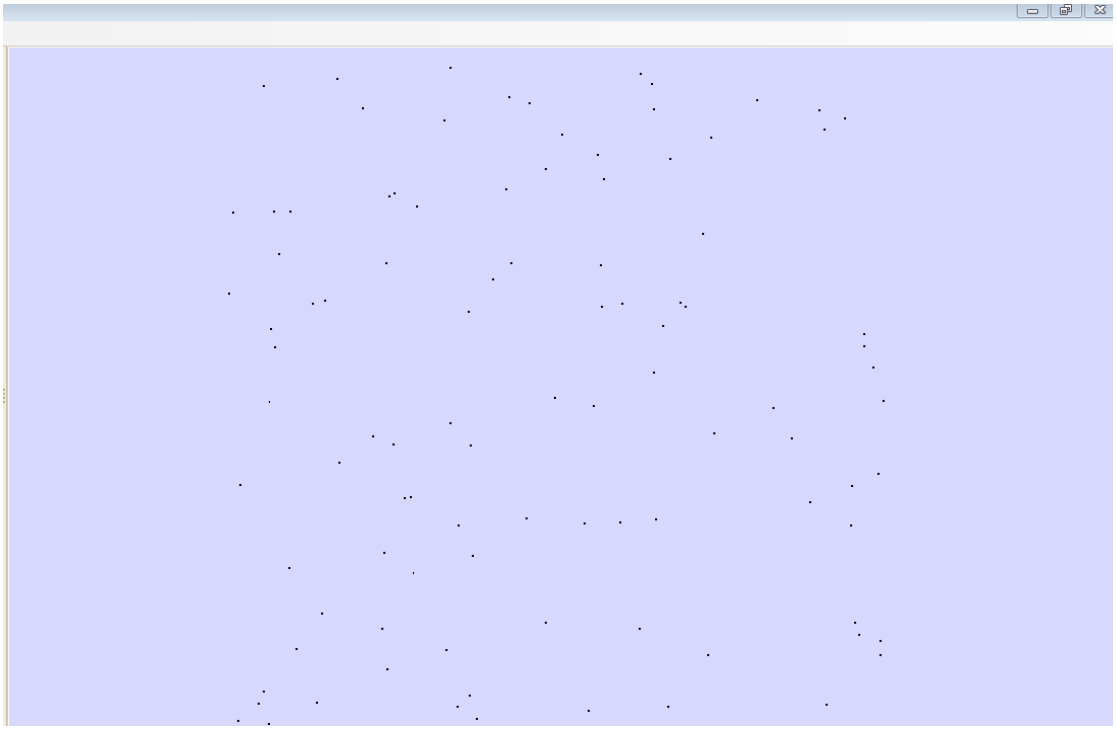
5.1 2-boyutlu Standart Voronoi Diyagramı ve Delaunay Üçgenlemesi

İki boyutlu uzayda verilen bir nokta kümesinin Voronoi diyagramını ve Delaunay üçgenlemesini oluşturulmasını sağlayan bölümdür. Nokta kümesi program içinde rastgele üretilebileceği gibi bir dxf formatlı dosyadan da okunabilmektedir. Rastgele bir nokta kümesi üretmek için 'Nokta Kümesi Oluştur' butonuna basılarak Nokta Üreteci penceresi açılır. Buradan gerekli parametreler girilerek noktalar ekranda görüntülenir. (Şekil 5.1 ve Şekil 5.2)

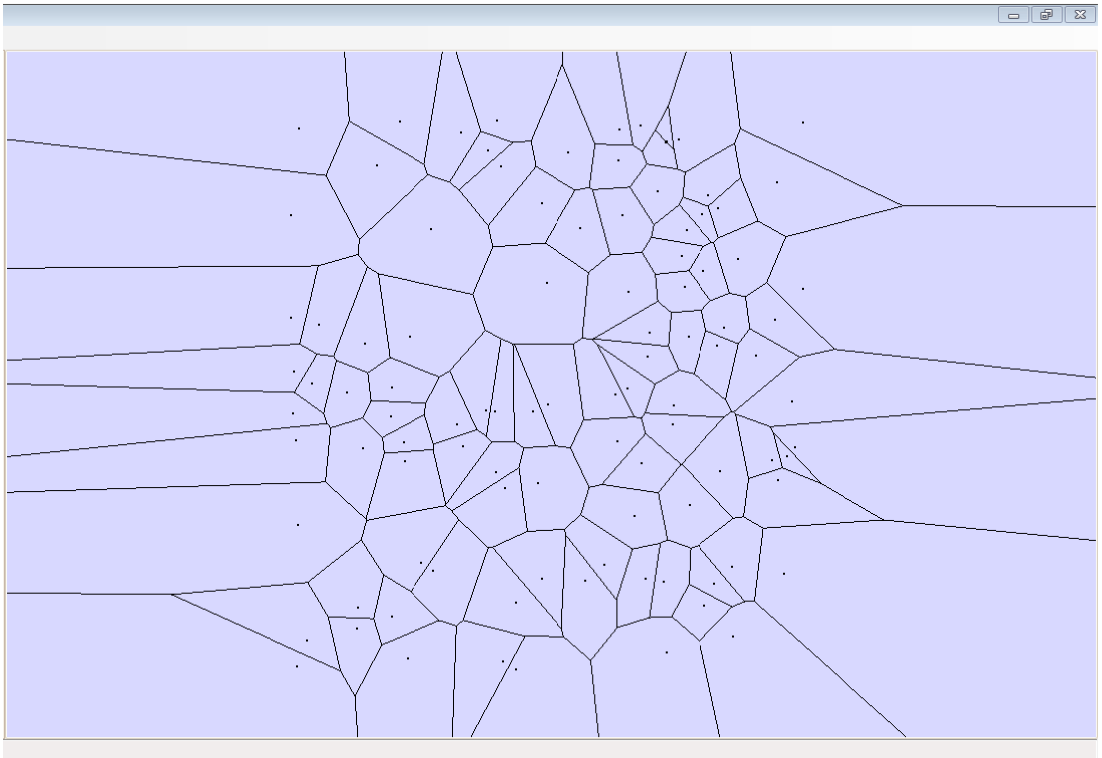


Şekil 5.1 Nokta üreteci ekranı

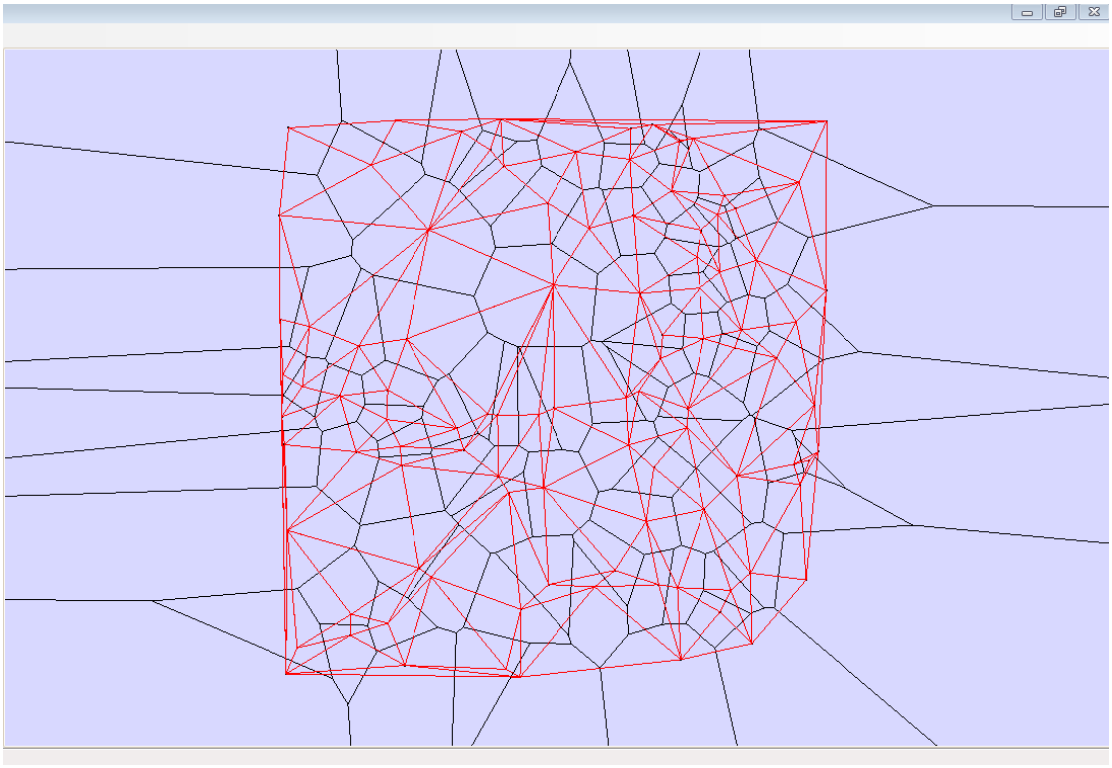
Bundan sonra bu nokta kümesi için Voronoi diyagramı (Şekil 5.3) ve Delaunay üçgenlemesi (Şekil 5.4) hesaplanarak görüntülenebilir.



Şekil 5.2 Rastgele üretilmiş 100 adet nokta



Şekil 5.3 Rastgele üretilmiş noktaların Voronoi diyagramı



Şekil 5.4 Rastgele üretilmiş noktaların Voronoi diyagramı ve Delaunay üçgenlemesi.

Voronoi diyagramının ve Delaunay üçgenlemesinin hesaplanması işlemi için program dahilinde çeşitli algoritmalar kullanılmış ve aynı bilgisayar üzerinde değişik nokta sayılarına göre Fortune algoritmasının çalışma süresi ölçülmüştür. Elde edilen sonuçlar Çizelge 5.1'de verilmektedir. Test için kullanılan bilgisayar, AMD Turion64 TL-58 1.9GHz işlemcili 2GB RAM'e sahip bir bilgisayardır.

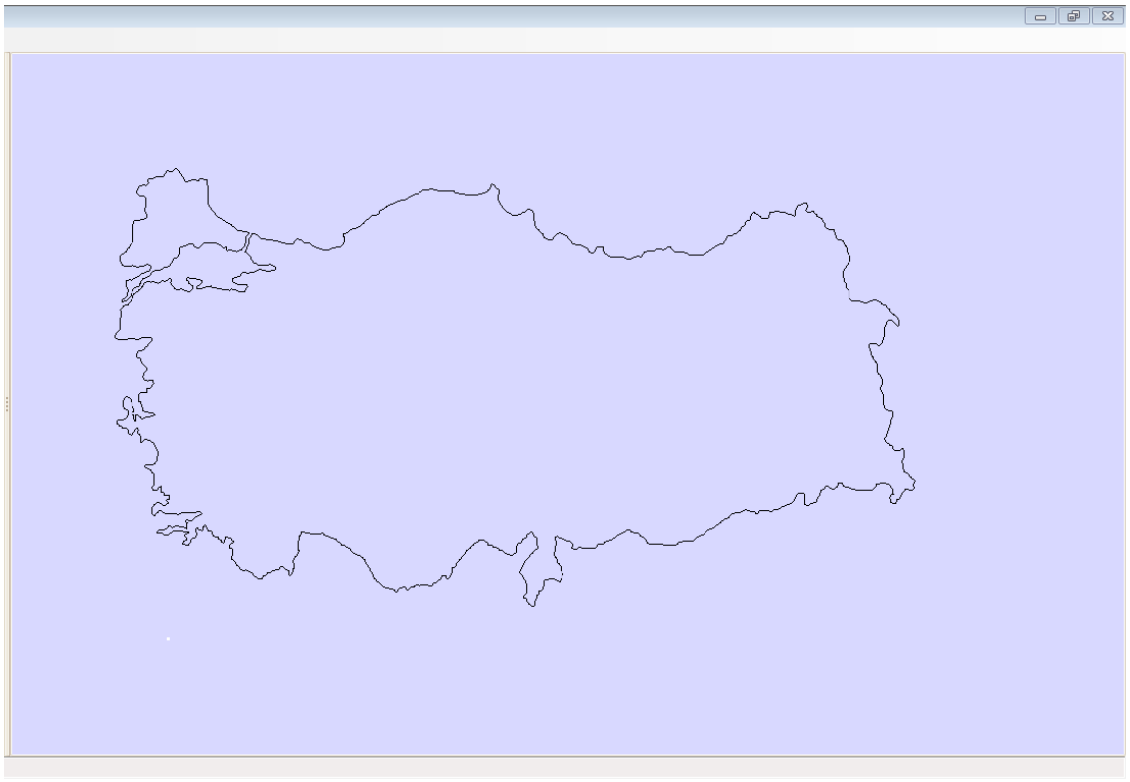
Çizelge 5.1 Test nokta kümeleri için CPU çalışma süreleri

	CPU çalışma süresi (ms)
Nokta Sayısı	Fortune
10	1.7136
100	2.1871
1000	7.7608
10000	87.8003

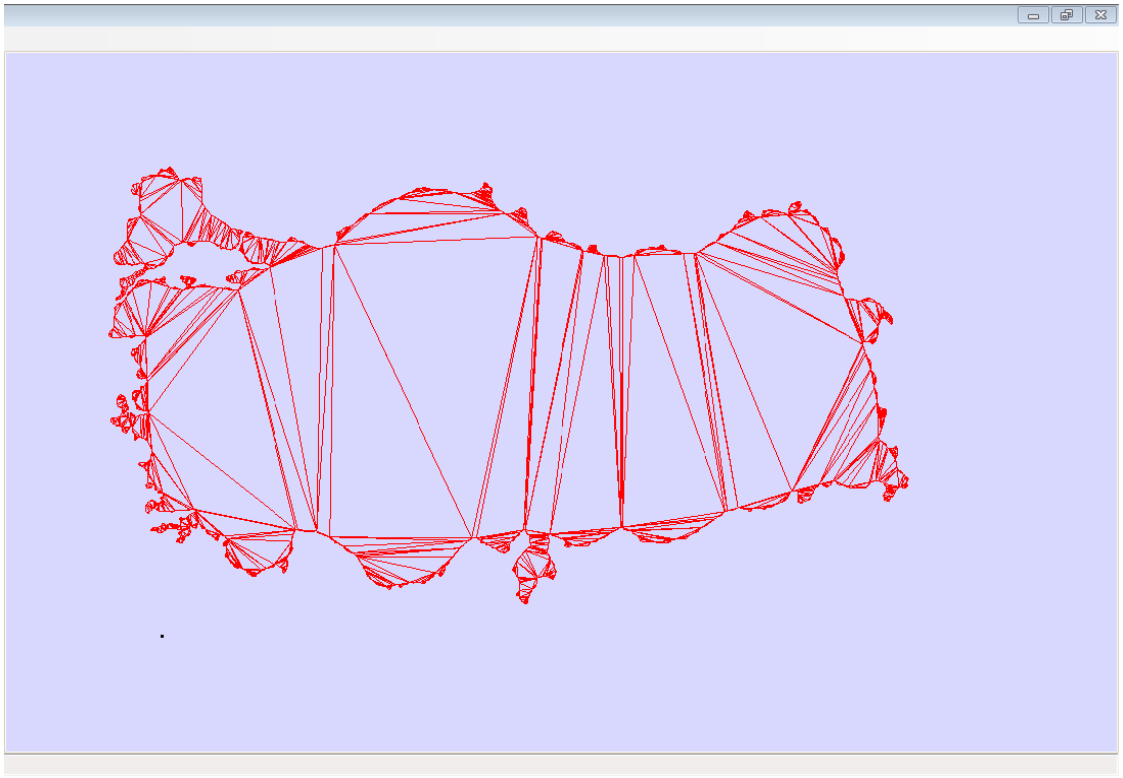
5.2 Çeşitli Voronoi Diyagramı ve Delaunay Üçgenlemesi Genelleştirmeleri

İki boyutlu kısımda yapılabilen diğer işlemlerin başında verilen bir konveks ya da konkav poligonal bölgenin üçgenlenmesi gelmektedir. Poligonun oluşturulması için program içindeki "Poligon üreteci" kullanılabileceği gibi dxf formatında kaydedilmiş bir poligon da programa alınabilir. Bundan sonra bu poligonun üçgenlemesi gerçekleştirilebilir. Yine girilen bir poligonun iç ekseni de hesaplatılıp görselleştirilebilir.

Şekil 5.5'te Türkiye sınırlarının oluşturduğu poligonlar gösterilmiştir. Türkiye'nin coğrafi konumu gereği sınırları iki poligondan oluşmaktadır. Bu poligonlar bir dxf dosyasından alınmıştır ve toplam 2101 nokta içermektedir. Şekil 5.6'da ise bu poligonların kısıtlanmış Delaunay üçgenlemesinin hesaplanması ile elde edilen görüntü sunulmaktadır. Bu poligonları oluşturan noktaların Delaunay üçgenlemesi hesaplandığında toplam 4154 adet üçgen elde edilmektedir ancak kısıtlanmış Delaunay üçgenlemesinde ise 2097 adet üçgen mevcuttur. Diğer üçgenler ise kısıtlamayı oluşturan sınırların dışında kalmaktadır.



Şekil 5.5 Türkiye sınırlarını gösteren poligon



Şekil 5.6 Türkiye sınırlarının Delaunay üçgenlemesi

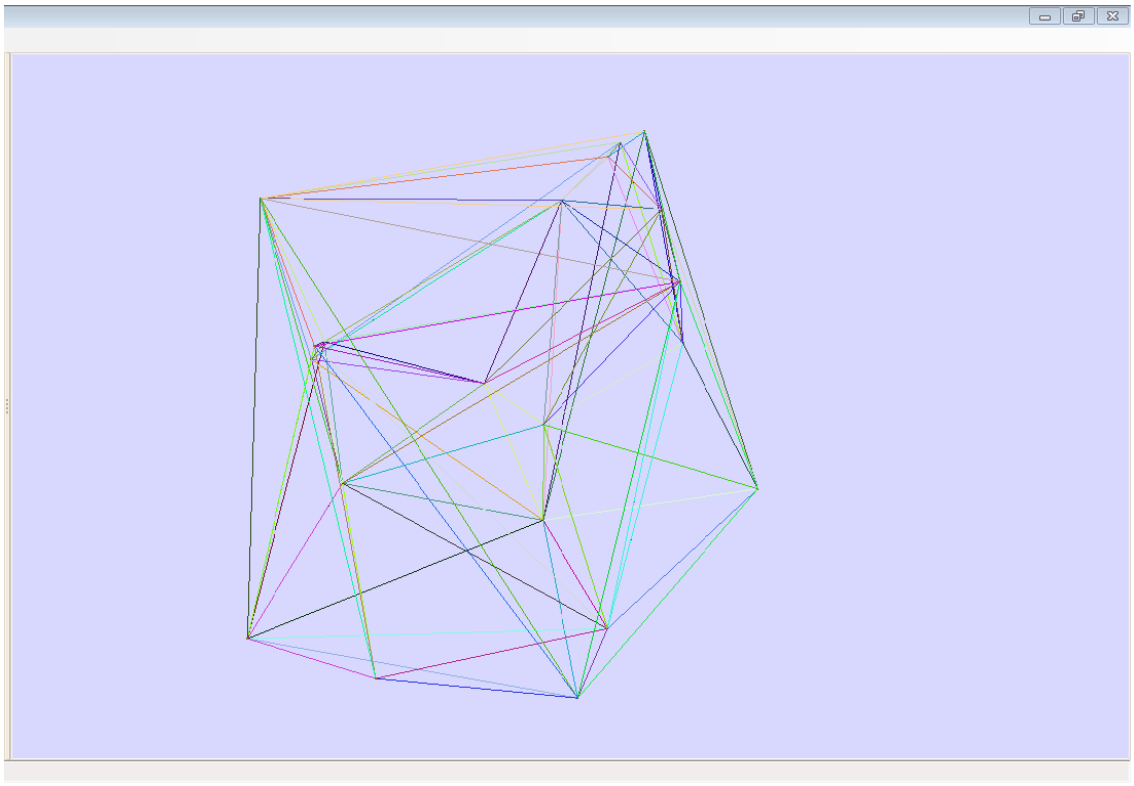
Program içinde yine verilen bir poligon için iç eksen hesaplaması yapılabilmektedir. Türkiye sınırlarını gösteren poligon için elde edilen iç eksen görüntüsü Şekil 5.7'de verilmiştir.

Bunun yanında bu kısımda, verilen bir nokta kümesi için -şekli ve -iskeleti bulunabilmektedir. Bu işlemlerde ve parametreleri değiştirilerek sonuçlar ekranda görüntülenebilir.

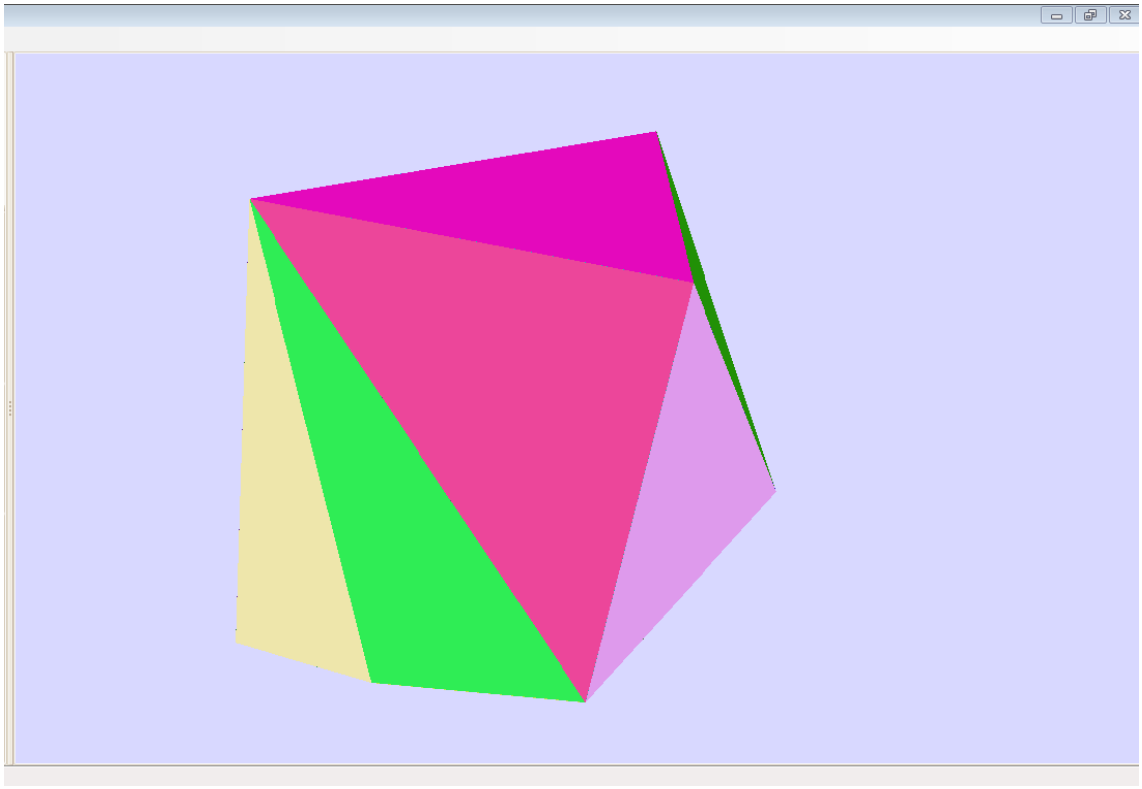
Şekil 5.7 Türkiye sınırları için hesaplanan iç eksen

5.3 Üç Boyutlu Uzayda Delaunay Üçgenlemesi

Programda 3D sekmesine geçilerek oluşturulan örnek 3-boyutlu nokta kümesi için Delaunay üçgenlemesi hesaplanabilmektedir. 3-boyutlu uzayda orijinden maksimum 10 birim uzaklıkta rastgele üretilmiş 20 adet noktanın Delanay üçgenlemesi Şekil 5.8'de gösterilmektedir. İstenirse oluşturulan üçgen yüzler boyanabilir (Şekil 5.9). Bu durumda elde edilen görüntü ilgili nokta kümesinin konveks zarfına karşılık gelmektedir.



Şekil 5.8 Üç boyutlu uzayda Delaunay Üçgenlemesi



Şekil 5.9 Üç boyutlu uzayda yüzlerin boyanması ile elde edilen Delaunay Üçgenlemesi

6. SONUÇLAR ve GELECEK ÇALIŞMALAR

Bu tez çalışmasında başlangıçta belirlenen amaçlar doğrultusunda Voronoi Diyagramı ve Delaunay üçgenlemesi yapılarını hesaplayıp görselleştirebilen bir yazılım geliştirilmeye çalışılmıştır. Bu yazılım C# programlama dilinde yazılmış olup bazı Matlab fonksiyonları da MATLAB Builder NE araç kutusu vasıtasıyla programa dahil edilerek kullanılmıştır.

Bu yapıların görselleştirilmesi için önce standart Windows GDI kütüphaneleri kullanılmış, ancak elde edilen sonuçların yetersizliği ve esnek olmayışı nedeniyle OpenGL kütüphaneleri kullanılmasına karar verilmiştir. Bunun için de C# dili için OpenGL kütüphanelerini kapsayan ve açık kaynak kodlu olan SharpGL yazılımı kullanılmıştır.

Program dahilinde öncelikle Voronoi diyagramlarının ve Delaunay üçgenlemesinin hesaplanması için optimal kabul edilen bazı algoritmalar geliştirilmeye çalışılmıştır. Bu algoritmalardan Fortune algoritması olarak da bilinen tarama çizgisi algoritması geliştirilmiş ve başarılı bir şekilde çalıştırılmıştır. Yine rastgele artırımlı algoritmalardan Lawson algoritması da yazılmış ve bu algoritmayla da sonuçlar elde edilmiştir. Böylece bu algoritmaların performans değerlendirilmesi imkanı da elde edilmiştir.

Bazı Voronoi diyagramı ve Delaunay üçgenlemesi genelleştirmeleri de program dahilinde geliştirilmiş ve görselleştirilmiştir. Bunun için saf C# kodu yazmak yerine bazı Matlab fonksiyonları kullanılmıştır.

Geliştirilen yazılım iki boyutlu noktalar kümelerinin ve iki boyutlu poligonal bölgelerin Voronoi diyagramı ve Delaunay üçgenlemesini başarılı bir şekilde gerçekleştirmektedir. Yine α-şekilleri, β-iskeletleri ve iç eksen hesaplaması yapılabilmekte ve görselleştirilebilmektedir. Ancak henüz üç boyutlu uzayda bu işlemler sağlanamamıştır.

Bu tez çalışması Voronoi diyagramı ve Delaunay üçgenlemesi ile ilgili çalışma ve araştırma yapacak araştırmacılara temel düzeyde bilgi sunmaktadır. Geliştirilen yazılım ise başta hesaplamalı geometri alanı olmak üzere coğrafik bilgi sistemleri, hareket planlama, yüzey modelleme gibi birçok alanda yararlı bir araç olarak kullanılabilecek niteliktedir.

Bu tez çalışmasının ve yazılımın devamı olarak yüzey modelleme ve örgü üretimi algoritmalarının da incelenmesi ve yazılıma dahil edilmesi hedeflenmektedir.

KAYNAKLAR

Aurenhammer, F. and Klein, R. 2000. Voronoi Diagrams, in: Handbook of Computational Geometry. Sack, J.R., Urrutia, J., Elsevier Science B.V., 201-290, Amsterdam.

Bowyer, A. 1981. Computing Dirichlet tessellations, The Computer Journal, 24(2), 162–166.

Chen, J. 1996. Computational Geometry: Methods and Applications. Lecture Notes, Computer Science Department, Texas A&M University, 227 p., U.S.A.

Anonymous. 2009. Web Sitesi. http://cgal.org. Erişim Tarihi: 10.03.2010.

de Berg, M. Cheong, O., van Kreveld, M., Overmars, M. 2008. Computational Geometry-Algorithms and Applications 3rd Edition. Springer, 386 p., Berlin.

Delaunay, B. 1934. Sur la sphere vide. A la memoire de Georges Voronoi, Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk 7, 793-800.

Dirichlet, PG.L. 1850. Uber die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen, Reine Angew. Math. 40, 209-227.

Domiter, V. and Zalik, B. 2008. Sweep-line algorithm for constrained Delaunay triangulation. International Journal of Geographical Information Science, 22(4), 449-462.

Dwyer, R.A. 1991. Higher-dimensional Voronoi diagrams in linear expected time. Discrete Comput. Geom. 6, 343-367.

Edelsbrunner, H. 1987. Algorithms in Combinatorial Geometry, EATCS Monographs on Theoretical Computer Science 10, Springer-Verlag, West Germany.

Edelsbrunner, H. and Mücke, E.P. 1994. Three-dimensional alpha shapes, ACM Trans. Graph. 13(1), 43-72.

Edelsbrunner, H., Kirkpatrick D.G., Seidel, R. 1983. On the shape of a set of points in the plane, IEEE Trans. Inform. Theory IT-29, 551-559.

Eppstein, D. 2009. Nearest Neighbors and Voronoi Diagrams. Web Sitesi http://www.ics.uci.edu/~eppstein/junkyard/nn.html. Erişim Tarihi: 10.03.2010.

Fortune, S. J. 1987. A sweepline algorithm for Voronoi diagrams. Algorithmica, 2,153–174, New York.

Fortune, S. 2004. Voronoi Diagrams and Delaunay Triangulations, In: Handbook of Discrete and Computational Geometry. Rosen, K.H., C.R.C. Press, Section 23, U.S.A.

Gabriel, K.R. and Sokal, R.R. 1969. A new statistical approach to geographic variation analysis. Systematic Zoology 18, 259-278.

Green, P.J. and Sibson, R.R. 1978. Computing Dirichlet tessellations in the plane, Comput. J. 21, 168-173.

Guibas, L. and Stolfi, J. 1985. Primitives for the Manipulation of General Subdivisions and the Computations of Voronoi Diagrams. ACM Trans. Graphics 4, 74-123.

Guibas, L.J., Knuth, D.E., Sharir, M. 1992. Randomized incremental construction of Delaunay and Voronoi diagrams, Algorithmica, 381-413.

Jaromczyk, J.W., Kowaluk, M., Yao, F. 1992. An optimal algorithm for constructing β-skeletons in the l_p-metric, SIAM J. Comput.

Kao, T.C. and Mount, D.M. 1992. Incremental construction and dynamic maintenance of constrained Delaunay triangulations, Proc. 4th Canad. Conf. Comput. Geom., 170-175.

Key, T.K. 2007. Curve and Surface Reconstruction. Cambridge University Press, 230 p., New York.

Kirkpatrick, D.G. 1979. Efficient computation of continuous skeletons, Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci., 18-27.

Kirkpatrick, D.G. and Radke, J.D. 1985. A framework for computational morphology. Computational Geometry, G.T. Toussaint, ed., North-Holland, Netherlands, 217-248.

Lawson, C.L. 1977. Software for C^1 surface interpolation. Math. Software IE, J.R. Rice, ed.. Academic Press, New York, 161-194.

Lee, D.T. 1982. Medial axis transformation of a planar shape, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-4, 363-369.

Lingas, A. 1994. A linear-time construction of the relative neighborhood graph from the Delaunay triangulation, Comput. Geom. 4, 199-208.

Anonymous. 2009. Matlab Builder NE User's Guide. Mathworks Inc., U.S.A.

Matula, D.W. and Sokal, R.R. 1980. Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane, Geogr. Anal. 12, 205-222.

Muller, D.E. and Preparata, F.P. 1978. Finding the intersection of two convex polyhedra, Theoret. Comput. Sci. 7, 217-236.

O'Rourke, J. 1997. Computational Geometry In C Second Edition. Cambridge University Press, 376 p., U.S.A.

Okabe, A., Boots, B., and Sugihara, K. 2000. Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd ed. Wiley, 532p., New York.

Anonymous. 1995. Web Sitesi. http://www.qhull.org. Erişim Tarihi: 10.03.2010.

Shamos, M.I. and Hoey, D. 1975. Closest-point problems, Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci., 151-162.

Su, P. and Drysdale, R.L.S. 1995. A comparison of sequential Delaunay triangulation algorithms, Proc. 11th Annu. ACM Sympos. Comput. Geom., 61-70.

Anonymous. 1994. Web Sitesi. http://www.mathworks.com. Erişim Tarihi: 10.03.2010.

Anonymous. 2008. Web Sitesi. http://www.voronoi.com. Erişim Tarihi: 10.03.2010.

Voronoi, G.M. 1908. Nouvelles applications des parametres continus a la theorie des formes quadratiques. deux-ieme Mimoire: Recherches sur les parallelloedres primitifs, J. Reine Angew. Math. 134, 198-287.

Voronoi, G.M. 1909. Deuxieme memoire: recherches sur les paralleloedres primitifs, J. Reine Angew. Math. 136, 67-181.

Watson, D.F. 1981. Computing the *n*-dimensional tessellation with application to Voronoi polytopes, The Computer Journal, 24(2), 167–172.

Anonymous. 2008. Web Sitesi. http://www.wolfram.mathworld.com. Erişim Tarihi: 10.03.2010.

Zimmer, H. 2005. Voronoi and Delaunay Techniques. Lecture Notes, Computer Sciences VIII, 14 p., Aachen.

EKLER

EK 1 OPENGL (OPEN GRAPHICS LIBRARY)

EK 2 MATLAB BUILDER NE

EK 1 OPENGL (OPEN GRAPHICS LIBRARY)

OpenGL için bir tanım yapmak gerekirse grafik donanımı için yazılmış bir yazılım ara yüzüdür. Farklı bir ifadeyle taşınabilir ve oldukça hızlı bir 3-boyutlu grafik ve modelleme kütüphanesidir. OpenGL Silicon Graphics, Inc. (SGI) tarafından geliştirilmiş ve optimize edilmiş algoritmalar kullanmaktadır. OpenGL bir programlama dili değildir, önceden paketlenmiş C dili çalışma zamanı kütüphanelerinden oluşur (Richard et.al. 2007).

OpenGL günümüzde CAD ve mimari yazılımlarından, modelleme programlarına oyunlara ve hatta sinema filmlerine kadar birçok amaç için kullanılmaktadır. Donanım hızlandırıcı kartlar ve hızlı mikroişlemciler sayesinde 3-boyutlu grafikler sadece oyunlar ve bilimsel çalışmalarda kullanılmaktan çıkmış, birçok tüketici programında tipik öğeler haline gelmiştir.

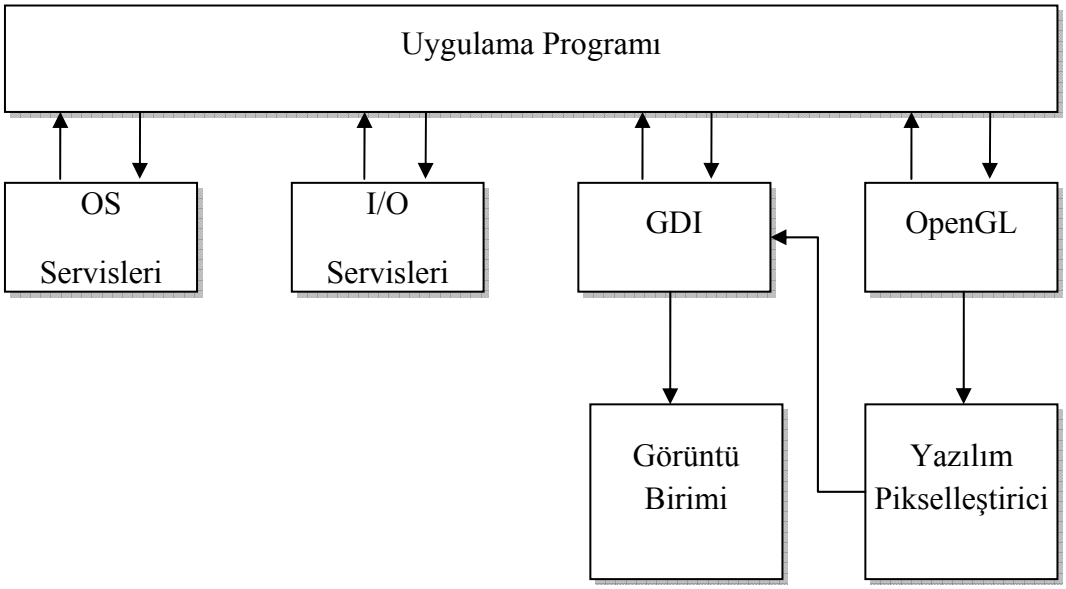
OpenGL prosedürel bir grafik API'dir. Programcı, sahnenin nasıl olduğunu betimlemek yerine belli bir görüntü ya da efekti elde etmek için gerekli adımları sıralar. Bu adımlar çeşitli OpenGL komutlarına çağrılarla gerçekleştirilir. Bu komutlar üç boyutta nokta çizgi poligon gibi temel grafik öğelerinin çizilmesini sağlar. Bunun yanında, OpenGL aydınlatma ve gölgelendirme, doku giydirme, renk karıştırma, transparanlık, animasyon ve daha birçok özel efekti desteklemektedir.

Genel Uygulamalar

Genel uygulama bir yazılım uygulamasıdır. Donanım uygulamaları ise grafik kartı veya oyun konsolu gibi belli bir donanım birimi için oluşturulmuş uygulamalardır. Genel bir uygulama teknik olarak sistem üretilen grafik resmini görüntüleyebildiği sürece hemen her platformda çalıştırılabilmektedir.

Şekil 1'de bir Windows programı çalışırken OpenGL ve genel uygulamanın görev aldığı yer gösterilmektedir. Tipik bir program, bazıları programcı tarafından yazılan,

bazıları da işletim sistemi yada programlama dili çalışma zamanı kütüphaneleri tarafından sağlanan birçok fonksiyon çağırır. Ekranda bir çıktı oluşturmak isteyen Windows programları genellikle GDI (*Graphics Device Interface*) olarak adlandırılan bir Windows API'ye çağrıda bulunur. GDI, bir pencerede yazı yazmayı, basit iki boyutlu çizimler yapmayı ve buna benzer işlemleri yerine getirmeyi sağlayan metotlar içermektedir. Genellikle grafik kartı üreticileri işletim sisteminin monitör üzerinde çıktı oluşturmasını izin veren donanım sürücüleri sağlar. OpenGL'in yazılım uygulaması bir uygulamadan grafik isteklerini alır ve 3-boyutlu grafiğin bir resmini oluşturur. Sonra bu resim monitörde görüntülenmek üzere işletim sistemine gönderilir. Diğer işletim sistemlerinde de süreç hemen hemen aynıdır, sadece GDI yerine işletim sisteminin yerel grafik servisleri yer almaktadır.



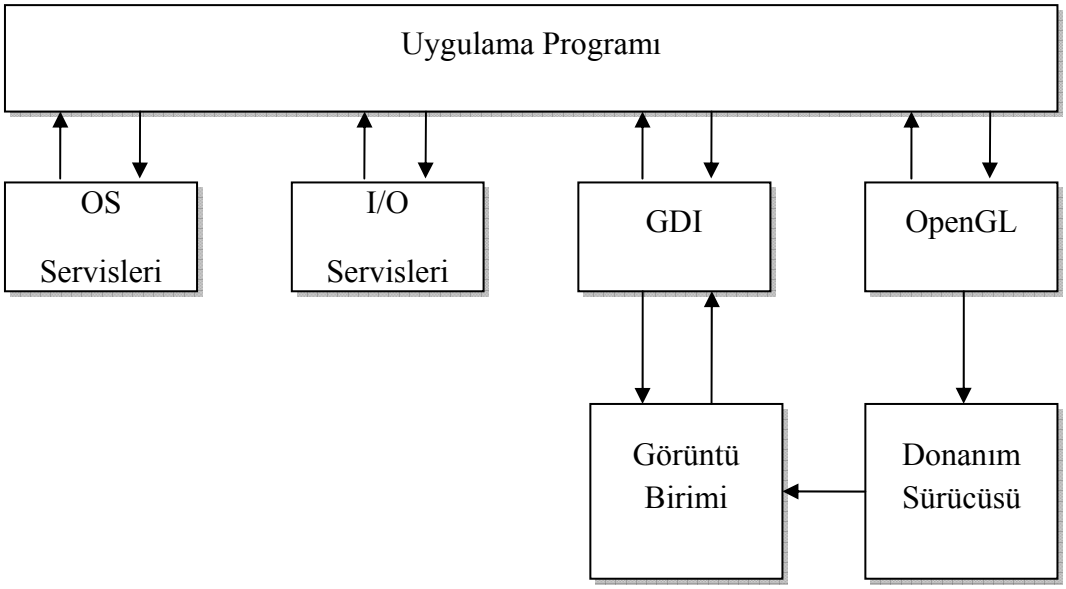
Şekil 1 OpenGL yazılım uygulaması

OpenGL çeşitli genel uygulama versiyonlarına sahiptir. Microsoft Windows NT 3.5 ve Windows 95(SR2)'den bu yana OpenGL genel uygulamasını işletim sistemi ile birlikte sağlamaktadır.

Donanım Uygulamaları

OpenGL'in donanım uygulaması genellikle bir grafik kartı sürücüsü halinde olmaktadır. Şekil 2'de uygulama programıyla OpenGL donanım uygulamasının ilişkisi gösterilmektedir. OpenGL API çağrıları bir donanım sürücüsüne iletilmektedir. Bu sürücü çıkış bilgilerini Windows GDI'ya göndermez, doğrudan grafik görüntüleme donanımına arabirim olarak çalışır.

Bir donanım uygulaması sıklıkla hızlandırılmış uygulama olarak adlandırılır çünkü donanım destekli 3-boyutlu grafikler yazılım uygulamalarına göre çok daha yüksek performans sağlar.



Şekil 2 OpenGL donanım uygulaması

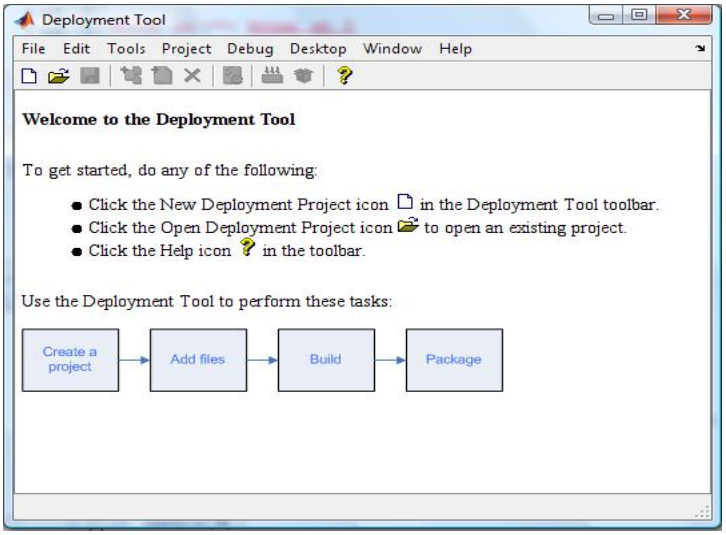
EK 2 MATLAB BUILDER NE

"Matlab Builder NE", Matlab fonksiyonlarının paketlenmesini ve .NET programcılarının bu fonksiyonları, CLS uyumlu herhangi bir dilde kullanabilmesini sağlayan bir "MATLAB Compiler" eklentisidir. Bu araç, Matlab fonksiyonlarını, Matlab kodları içeren .NET metotlarına çevirmektedir (Anonymous 2009).

Builder NE ile oluşturulan .NET elemanları bir ya da birden çok sınıf içerir. Bu sınıflarda derleme aşamasında projeye eklenen Matlab m-fonksiyonlarına ulaşılmasını sağlar. Matlab veri tipleriyle çalışılabilmesi için "MWArray" veri tipi dönüşüm sınıfı sunulmuştur. Derlemeye eklenecek m-kodları muhakkak m-fonksiyonu formatında olmalıdır.

Matlab Builder NE Kullanımı

Öncelikle yeni bir m-dosyası açılıp derlemeye eklenecek Matlab kodu fonksiyon biçiminde yazılır ve fonksiyon ismi ile kaydedilir. Matlab komut satırına "deploytool" yazılarak veya Matlab start menüsünden "deploytool" çalıştırılır (Şekil 1). Yeni proje oluşturularak, proje tipi olarak ".NET Component" seçilir. Projede gerekli görülen sınıflar oluşturulup, isimleri düzenlenir. Daha sonra ister sürükle-bırak ile ister "Add File" butonu ile oluşturulan m-fonksiyonları sınıflara dâhil edilir. Derleme butonuna basılarak derleme işlemi gerçekleştirilir.



Şekil 1 Deploytool ekranı

MWArray Veri Tipi Dönüşüm Sınıfı

Aşağıdaki tabloda .NET veri tiplerine karşılık hangi Matlab veri tiplerinin kullanılması gerektiği ve dönüşüm için kullanılması gereken sınıf çizelge 1'de verilmiştir.

Çizelge 1 MWArray veri tipleri

.NET Tipi	MWArray Dönüşüm Tipi	MATLAB Tipi
System.Double	MWNumericArray	double
System.Number	MWNumericArray	double
System.Float	MWNumericArray	single
System.Byte	MWNumericArray	int8
System.Short	MWNumericArray	int16
System.Int32	MWNumericArray	int32
System.Int64	MWNumericArray	int64
System.Char	MWCharArray	char
System.String	MWCharArray	char
System.Boolean	MWLogicalArray	logical
N/A	MWStructArray	structure
N/A	MWCellArray	cell

ÖZGEÇMİŞ

Adı Soyadı : Bülent ÖZBEK

Doğum Yeri : Kastamonu

Doğum Tarihi : 07.08.1977

Medeni Hali : Bekar

Yabancı Dil : İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : Mustafa Kaya Anadolu Lisesi (1995)

Lisans : Karadeniz Teknik Üniversitesi Mühendislik-Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği Bölümü (1999)

Yüksek Lisans : Ankara Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalı (Mart 2010)

Çalıştığı Kurum/Kurumlar ve Yıl

Eti Bakır A.Ş. Genel Müdürlüğü Elektronik Mühendisi 2000-2001.

Ankara Üniversitesi Kastamonu M.Y.O. Öğretim Görevlisi 2001-2007

Kastamonu Üniversitesi Kastamonu M.Y.O. Öğretim Görevlisi 2007-.